HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Alrik Messner

Betweenness Clusterung für schlichte, ungerichtete Graphen

2012

Fakultät **Mathematik/Naturwissenschaften/Informatik**

BACHELORARBEIT

Betweenness Clusterung für schlichte, ungerichtete Graphen

Autor:
Alrik Messner

Studiengang:
Angewandte Mathematik

Seminargruppe:
MA08w1

Erstprüfer:
Prof. Dr. Peter Tittmann

Zweitprüfer:
Prof. Dr. Eckhard Manthei

Mittweida, Februar 2012

Bibliografische Angaben

Messner, Alrik: Betweenness Clusterung für schlichte, ungerichtete Graphen, 73 Seiten, 12 Abbildungen, Hochschule Mittweida (FH), Fakultät Mathematik/Naturwissenschaften/Informatik

Bachelorarbeit, 2012

Referat

Es ist möglich, Graphen und Netzwerke durch Bewertung der Kanten mit Hilfe des Zentralitätsindizes Betweenness in Cluster zu zerlegen. Die Berechnung der Betweennesswerte für jede Kante eines betrachteten Graphen benötigt eine Zeit von $O(n^2m)$ für $m >> n$. In dieser Arbeit wird eine schnellere Methode mit einer Zeitkomplexität von $O(nm)$ für die Berechnung eines Betweenness Rankings nach Newman und unabhängig nach Brandes vorgestellt und implementiert. Es wird ein Clusteralgorithmus nach Newman und Girvan auf Basis des Index Kanten-Betweenness und mit einer Laufzeit von $O(nm^2)$ vorgestellt und es werden verschiedene Graphen damit geclustert. Die Arbeit ist restringiert auf schlichte, ungerichtete Graphen.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

1 Einleitung

Zu den wesentlichen Bestandteilen des menschlichen Denkens gehört die Eigenschaft, Objekte unserer Anschauung anhand ihrer Eigenschaften als zusammengehörig oder aber auch als nicht zusammengehörig zu erkennen. Dabei werden diese Objekte aus der Gesamtheit der Objekte heraus in Teilmengen von Objekten mit gleichen beziehungsweise ähnlichen Eigenschaften unterteilt. Eine derartige Unterteilung kann grundlegend auf zwei Arten entstehen:

- Klassifikation und
- Clusterung.

Bei einer Klassifikation werden die zu bildenden Teilmengen, Klassen genannt, bereits vorgegeben und die Objekte sind anhand ihrer Eigenschaften diesen Klassen zuzuordnen. Dabei kann es durchaus sinnvoll sein, ein Objekt auch mehreren Klassen zuzuordnen. Nehmen wir als Grundmenge die Menge aller Fahrzeuge an. Nehmen wir weiter folgende Klassen als gegeben an:

1. motorisierte Fahrzeuge
2. nicht motorisierte Fahrzeuge
3. rote Fahrzeuge und
4. PKW.

Eigenschaften wären dabei zum Beispiel die Farbe und die Motorisiertheit des Fahrzeugs. Es ist leicht zu erkennen, dass hier tatsächlich ein rotes Fahrrad sowohl in der Klasse der nicht motorisierten Fahrzeuge als auch in der Klasse der roten Fahrzeuge liegt. Gleichzeitig ist die Klasse der PKW eine Unterklasse der Klasse der motorisierten Fahrzeuge. Es ist also möglich, Klassen als neue Grundmenge aufzufassen und diese nochmals weiter zu klassifizieren.

Im Gegensatz zur Klassifizierung werden beim Clustern keine Klassen vorgegeben. Anhand der Eigenschaften der Objekte, werden die Klassen, hier Cluster genannt, erst während des Klassifizierungsprozesses gebildet. Gegebenenfalls ist dann nach dem Clusterbildungsprozess eine Interpretation der gebildeten Cluster notwendig.

2 Clusterung von Graphen

2.1 Grundlegende Definitionen und Vereinbarungen

Diese Arbeit beschäftigt sich mit dem Clustern von Graphen. Als erstes ist der hier verwendete Graphenbegriff zu präzisieren.

Definition 1 (Graph) *Ein schlichter, ungerichteter Graph $G = (V,E)$ hat die Knotenmenge $V = \{1,2,...,n\}$ und die Kantenmenge $E \subseteq \{\{u,v\},\ u \neq v,\ u,v \in V\}$.*

Lässt man die Bedingung $\{u,v\} \in E \rightarrow u \neq v$ für die Kanten fallen, wären Kanten der Form $\{v,v\},\ v \in V$ erlaubt, sogenannte Schlingen. Wird E nun nicht als Menge, sondern als Multimenge angenommen, wäre es möglich, dass eine Kante $\{u,v\}$ mehrfach in E vorkommt. Man spricht dabei von parallelen Kanten. Besitzt ein Graph wie oben definiert weder Schlingen noch parallele Kanten, nennt man ihn schlicht. Eine weitere Verallgemeinerung wäre, Kanten nicht als ungeordnetes Paar $\{u,v\}$ zu betrachten, sondern als geordnetes Paar. Solche Kanten nennt man gerichtete Kanten. Für gerichtete Kanten gilt dann $(u,v) \neq (v,u)$. Ein Graph mit (ausschließlich) gerichteten Kanten heißt gerichteter Graph.

Vereinbarung 1 *Der Begriff Graph bezeichnet im Folgenden einen schlichten und ungerichteten Graphen.*

Weiter ist es möglich, den Kanten eines Graphen $G = (V,E)$ Gewichte zuzuordnen. Dies geschieht mithilfe einer Funktion $\omega : E \rightarrow \mathbb{R}^+$. Die Gewichte stehen zum Beispiel für die Stärke der durch die Kante $e \in E$ repräsentierten Verbindung. Die Begriffe Adjazent und Nachbarschaft müssen eingeführt werden.

Definition 2 (Adjazenz, inzidenz, Nachbarschaft) *Sei $G = (V,E)$ ein Graph. Zwei Knoten $u,v \in V$ heißen adjazent oder benachbart in G, wenn gilt:*

$$\{u,v\} \in E.$$

Ein Knoten $w \in V$ heißt zu einer Kante $\{u,v\} \in E$ inzident, wenn gilt:

$$u = w \vee v = w.$$

Die Nachbarschaft eines Knotens v in G ist die Menge:

$$\{u : u \text{ ist adjazent zu } v.\}$$

Wir vereinbaren folgende Bezeichnungen:

Definition 3 *Sei $G = (V,E)$ ein Graph. Seien $X,Y \in V$. Wir definieren:*

$$E(X,Y) := \{\{x,y\} : \{x,y\} \in E \wedge x \in X \wedge y \in Y\}$$

und

$$E(X) := E(X,X).$$

Definition 4 (Knotengrad, Blatt) *Sei $G = (V,E)$ ein Graph. Der Knotengrad $deg(v)$, $v \in V$ für schlichte, ungerichtete Graphen ist definiert durch:*

$$deg(v) = |E(\{v\},V)|.$$

Ein Knoten $t \in V$, für den gilt:

$$deg(t) = 1$$

heißt Blatt.

Es wird später der Graph benötigt, der entsteht, indem man eine Teilmenge der Knotenmenge eines Graphen als Knotenmenge des neuen Graphen betrachtet und alle Kanten, die zwischen den Knoten dieser Teilmenge verlaufen, als Kantenmenge für diesen neuen Graphen ansieht.

Definition 5 (Von U induzierter Untergraph) *Sei $G = (V,E)$ ein Graph und $U \subset V$. Der von U auf G induzierte Untergraph hat die Form:*

$$G[U] := (U, E(U)).$$

Des weiteren wird der Begriff der Clusterung eines Graphen benötigt.

Definition 6 (Clusterung eines Graphen) *Sei $G = (V,E)$ ein Graph und ferner sei $C = \{C_1, C_2, \ldots, C_k\}$ eine Menge von Teilmengen $C_i \subseteq V$, $C_i \neq \emptyset$ für die*

- $C_i \cap C_j = \emptyset$, $1 \leq i < j \leq k$ *und*
- $\bigcup_{i \in \{1,\ldots,k\}} C_i = V$ *gilt.*

Dann heißt C Clusterung des Graphen G. Die Mengen C_i heißen die Cluster dieser Clusterung.

Für diese Arbeit wird also eine Clusterung als Partition der Knotenmenge eines Graphen betrachtet. Verallgemeinerungen dieser Definition sind möglich. Es wäre zum Beispiel

denkbar, leere Cluster zuzulassen oder die Knoten nur mit einer bestimmten Wahrscheinlichkeit zu einem Cluster zuzuordnen. Für einen Graphen G mit gegebener Clusterung zerfällt die Kantenmenge von G in zwei disjunkte Klassen. Die erste Klasse ist die Klasse der Kanten, die nur Knoten innerhalb desselben Clusters verbinden. Die andere Klasse ist die der Kanten, die nur Knoten verschiedener Cluster miteinander verbinden. Definieren wir diese Klassen:

Definition 7 (Intra-Cluster-Kanten, Inter-Cluster-Kanten) *Sei $G = (V,E)$ ein Graph. Sei $C = \{C_1, C_2, \ldots, C_k\}$ eine Clusterung von G. Die Menge*

$$E(C) := \bigcup_{i \in \{1,\ldots,k\}} E(C_i)$$

heißt Menge der Intra-Cluster-Kanten. Die Menge

$$\overline{E(C)} := E \setminus E(C)$$

heißt Menge der Inter-Cluster-Kanten.

Für die Menge der Inter-Cluster-Kanten gilt:

$$\overline{E(C)} = \bigcup_{\substack{i \in \{1,\ldots,k\} \\ j \in \{1,\ldots,k\} \\ i \neq j}} E(C_i, C_j).$$

2.2 Konstruktion eines Graphen aus einer Problemstellung aus der Praxis

Gegeben sei eine Menge M von Objekten und eine darauf definierte Ähnlichkeitsfunktion $\Phi : M \times M \to \mathbb{R}^+$, die den Objekten je paarweise einen reellen Wert als Maß für deren Ähnlichkeit zuordnet. Gilt dabei $\Phi(a,b) = \Phi(b,a),\ a,b \in M$ kann daraus ein schlichter, ungerichteter Graph konstruiert werden. Eine Variante dafür ist folgende:

1. Für jedes der Objekte aus unserer Menge wird ein Knoten des Graphen erzeugt.
2. Der Graph wird als vollständiger Graph angenommen.
3. Der Kante zwischen zwei Knoten wird der Wert der Ähnlichkeitsfunktion zugeordnet, der zu den beiden Objekten gehört, für die diese Knoten erzeugt wurden.

Oft ist das Ähnlichkeitsmaß auch als Distanz aufzufassen. Dann sind zwei Objekte zum Beispiel ähnlich, wenn sie nahe aneinander liegen. Wenn eine derartige Ähnlichkeitsfunktion nur Werte in $\{0,1\}$ annimmt, kann der Graph als Graph ohne Kantengewichte modelliert werden, indem für je zwei Objekte $a,b \in M$ genau dann eine Kante erzeugt wird, wenn $\Phi(a,b) = 1$ gilt. Ein Beispiel dafür wäre folgendes soziales Netzwerk. Als

Netzwerk wird im Rahmen dieser Arbeit jedes System bezeichnet, dass sich als Graph modellieren lässt. Die notwendigen Daten für das soziale Netzwerk seien folgendermaßen gegeben:

- Die Menge M der Objekte sei eine Menge von Personen.
- Die Ähnlichkeitsfunktion Φ sei ein Maß für die Bekanntschaft einer Person a zu einer anderen Person b. Im einfachsten Fall gilt:

$$\Phi(a,b) = \begin{cases} 1 & \text{Person a kennt Person b (und umgekehrt)} \\ 0 & \text{a und b sind einander nicht bekannt.} \end{cases}$$

Der später eingeführte Zentralitätsindex Betweenness findet vor allem in der Untersuchung solcher sozialen Netzwerke Anwendung. Diese Arbeit beschränkt sich genau auf den hierbei entstehenden Fall eines schlichten, ungerichteten Graphen ohne Kantengewichte. Verallgemeinerungen sind aber möglich.

Sollte für die Ähnlichkeitsfunktion $\Phi(a,b) \neq \Phi(b,a),\ a,b \in M$ gelten, führt dies auf gerichtete Graphen.

2.3 Graphenclusterung

Anhand des im vorherigen Abschnitt 2.2 vorgestellten Problems wird nun klar, dass genau die Objekte derselben Klasse zuzuordnen sind, die dicht miteinander verbunden sind. Nach außen, das heißt zu Knoten, die nicht in derselben Klasse liegen, sollten jedoch nur schwache Verbindungen existieren.

Aufgabe in der Graphenclusterung ist es also, Teilmengen der Knotenmenge zu finden, die genau diese Eigenschaften haben. Am Ende soll die Knotenmenge des Graphen komplett in derartige (disjunkte) Teilmengen, die sogenannten Cluster, zerlegt sein. Dabei gibt es verschiedene Ziele für einen derartigen Clusterprozess. Man unterscheidet hierbei vor allem zwischen Graphenpartitionierung und Strukturerkennung (Finding Community Structure) [4]. Außerdem kann es verschiedene Nebenbedingungen, vor allem bezüglich der Anzahl der Cluster und deren Größe (Zum Beispiel Mächtigkeit der Knotenteilmengen) geben. Auch hier ist eine hierarchische Clusterstruktur möglich.

2.3.1 Graphenpartitionierung

Zum Zweck der Graphenpartitionierung soll die Knotenmenge des Graphen G in eine vorgegebene Anzahl k an Clustern zerlegt werden. Dabei sollen möglichst wenige Kanten zwischen den Clustern liegen. Diese Forderungen führen auf ein Optimierungs-

problem:

$$|\overline{E(C)}| \to \min,\ |C| = k.$$

Oft wird zusätzlich gefordert, dass die Cluster möglichst gleiche Mächtigkeit haben. Das Abschneiden einzelner Blätter ist dabei meist unerwünscht.

Ein Beispiel für Graphenpartitionierung aus der Technik: Gegeben sei ein elektrischer Schaltkreis. Dieser bestehe aus Bauteilen, die mit Leitungen miteinander verbunden sind. Weiterhin sollen diese Bauteile auf eine gegebene Anzahl k von gleichgroßen Platinen gelötet werden, so dass möglichst wenige Leitungen zwischen den Platinen verlaufen. Die Größe und Anzahl der Platinen sei so abgestimmt, dass auf jede Platine die (ungefähr) gleiche Anzahl Bauteile aufzulöten ist.

Die im Abschnitt 2.2 vorgestellten Objekte der Menge M sind in diesem Fall die Bauteile. Unsere Ähnlichkeitsfunktion Φ sei:

$$\Phi(a,b) = \begin{cases} 1 & \text{Bauteil } a \text{ und Bauteil } b \text{ sind mit einer Leitung verbunden, } a,b \in M \\ 0 & \text{sonst.} \end{cases}$$

Dann wird der Graph G konstruiert. Für jedes Bauteil a_i wird ein Knoten i erzeugt. Immer dann, wenn gilt $\Phi(a_i,a_j) = 1$, wird zwischen den Knoten i und j die Kante $\{i,j\}$ erzeugt.

Dann entspricht das Aufteilen der Bauteile auf die Platinen einer Partitionierung der Knotenmenge des Graphen G, die die Mächtigkeit der Schnittkantenmenge als Zielfunktion minimieren soll mit den Nebenbedingungen:

1. Es sollen genau k Cluster erzeugt werden
2. Die Cluster sollen die gleiche Größe haben

Die Bedingung 2 wird dabei oft abgeschwächt sein, so dass nur ungefähr die gleiche Größe verlangt wird.

Ein weiteres Beispiel, diesmal aus der Informatik, ist die Zerlegung eines Algorithmus in parallel ausführbare Teilalgorithmen [4]. Ziel ist dabei, den Algorithmus auf einem Parallelcomputer auszuführen und dessen Parallelität möglichst voll auszunutzen. Dabei repräsentieren die Knoten des zu erzeugenden Graphen G die nicht trennbaren Teile des Algorithmus. Eine Kante muss immer dann eingefügt werden, wenn ein Abschnitt des Algorithmus Informationen von einem vorangegangenen Abschnitt benötigt. Diese Beziehung ist dann natürlich gerichtet und das Problem führt auf einen gerichteten Graphen. Je nach Art des Algorithmus und auch der verwendeten parallelen Rechentechnik können unterschiedliche Nebenbedingungen notwendig sein. Oft wird das Ziel sein, den

Algorithmus in so viele parallel ausführbare Teilalgorithmen zu zerlegen, wie parallele Recheneinheiten zur Verfügung stehen und diese sollten in etwa die selbe Rechenzeit benötigen (bei Recheneinheiten mit gleicher Leistung).

2.3.2 Strukturerkennung in Graphen

Eine andere Zielsetzung für die Graphenclusterung besteht darin, bisher nicht bekannte Strukturen in den zugrundeliegenden Datensätzen aufzuzeigen. Ist bei der Graphenpartitionierung die Anzahl der zu erzeugenden Cluster vorgegeben, so kann für den Zweck der Strukturerkennung nur unzureichend vorhergesagt werden, wie viele Cluster am Ende erzeugt werden. Vielmehr ist die Anzahl der letztendlich erzeugten Cluster Teil des Ergebnisses des Clusterprozesses. Der bei der Graphenpartitionierung zugrunde liegende Zweck verlangte oft, unabhängig von der tatsächlichen Struktur des erzeugten Graphen, Cluster gleicher Mächtigkeit. Eine Festlegung der Clustergrößen im Sinne der Strukturerkennung ist jedoch nicht sinnvoll. Die Objekte in einem in der Natur vorkommenden Netz müssen nicht zwingend zu gleich großen Gruppen eng vernetzt beziehungsweise ähnlich sein. Mit der Festsetzung von Clustergrößen kann in diesem Fall also bereits Information über die tatsächliche Struktur des Netzwerks verfälscht sein.
Das Beispiel des sozialen Netzwerks aus Abschnitt 2.2 verdeutlicht das Prinzip der Strukturerkennung. Zur Wiederholung: Gegeben sei eine Menge M von Personen. Weiterhin sei mit $\Phi : M \times M \rightarrow \{0,1\}$ eine Funktion, die je zwei Personen a und b aus M zuordnet, ob diese sich gegenseitig bekannt sind, also:

$$\Phi(a,b) = \begin{cases} 1 & \text{Person a kennt Person b (und umgekehrt)} \\ 0 & \text{a und b sind einander nicht bekannt.} \end{cases}$$

Nun soll es die Aufgabe des Clusteralgorithmus sein, Gruppen von Personen (Teilmengen von M) zu finden, die sich im Idealfall alle untereinander kennen. Dieser Idealfall würde auf das Zerlegen eines Graphen in Cliquen führen. Allerdings sind die wenigsten Graphen komplett in vollständige Untergraphen K_r mit $r \geq 3$ zerlegbar. Auch für ein soziales Netzwerk ist eine solche Struktur nicht zu erwarten, auch wenn dort ein häufiges auftreten von Cliquen zu erwarten ist. Allgemeiner werden Personengruppen gesucht, in denen sich die Mehrzahl der Personen untereinander kennen. Diese Gruppen werden im Allgemeinen nicht die gleiche Mächtigkeit besitzen.
Zusätzlich wird gerade in sozialen Netzwerken eine Person meist mehreren Personengruppen angehören, deren Mitglieder die Mitglieder der anderen Gruppen großteils nicht kennen. Beispiele dafür sind Familie, Arbeitskollegen und Mitglieder des Sportvereins, den die betrachtete Person besucht.

2.4 Methoden zur Graphenclusterung

Zum Clustern von Graphen werden Methoden aus verschiedensten Bereichen der Mathematik verwendet. Neben Varianten, die eine Startclusterung schrittweise verbessern, gibt es auch solche, die in einem Schritt die zu erzeugende Clusterung finden. Auch Zufallsalgorithmen sind bekannt.

2.4.1 Splitting, Linking, Shifting

Die drei folgenden Operationen ordnen einer Clusterung C eines Graphen G eine neue Clusterung C' von G zu:

1. Splitting: Beim Splitting wird ein Cluster der Clusterung C in zwei oder mehr kleinere Cluster zerlegt.
2. Linking: Beim Linking werden zwei oder mehr Cluster der Clusterung C zu einem einzigen neuen Cluster zusammengefasst.
3. Shifting: Hier wird ein Knoten oder eine Menge von Knoten aus einem Cluster der Clusterung C in ein anderes Cluster der Clusterung C verschoben.

Die Operationen werden anhand globaler oder lokaler Kostenfunktionen durchgeführt, um sicherzustellen, dass C' eine bessere Clusterung darstellt als C. Wie eine Bewertung der Güte einer Clusterung eines Graphen durchgeführt werden kann, ist Thema im Abschnitt 2.5.

Im weiteren werden Splitting-Algorithmen betrachtet. Man kann Splitting-Algorithmen danach unterteilen, in wie viele Cluster das zu teilende Cluster in einem Splitting-Schritt zerteilt wird. Wird das Cluster in 2 Teile geteilt, spricht man von einer Bisektion. Wird es in mehr Teile geteilt, von einer Multisektion. Für die Zwecke dieser Arbeit wird folgendes definiert:

Definition 8 (k-Sektion) *Sei $G = (V,E)$ ein Graph. Sei $C = \{C_1, \ldots, C_k\}$ eine Clusterung von G. Dann bezeichnet man das Zerteilen von G in die k von den Clustern C_i, $i \in \{1, \ldots, k\}$ induzierten Untergraphen $G[C_i]$ als eine k-Sektion.*

Oft findet auch der Begriff Schnitt Anwendung.

Definition 9 (Schnitt) *Sei $G = (V,E)$ ein Graph. Sei $C = \{C_1, \ldots, C_k\}$ eine Clusterung von G. Dann heißt die Menge $Cut(G,C) := \overline{E(C)}$ der Inter-Cluster-Kanten auch (k-1)-Schnitt (kurz Schnitt) von G.*

Size$(Cut(G,C)) := |\overline{E(C)}|$ heißt Größe des Schnitts.

Jede k-Sektion induziert einen $(k-1)$-Schnitt und umgekehrt.

Eine weitere wichtige Unterteilung richtet sich danach, ob der Graph in einem einzigen Schritt in die gewünschte Anzahl von Clustern zerlegt wird, oder in mehreren Schritten. Eine Variante für eine Zerlegung in mehreren Schritten ist eine wiederholte Bisektion. Dafür wird folgender Algorithmus durchgeführt:

Algorithmus 1 (Iterierte Bisektion) *Eingabe ist ein Graph $G = (V,E)$. Sei $U \subseteq V$. Sei $k \leq |V|$ die gewünschte Anzahl an Clustern. Sei c die Anzahl der bisher erhaltenen Cluster. Sei C eine Menge, die die bisher erzeugten Cluster enthält.*

1. *Setze U gleich V, $c = 1$, $C = \{V\}$.*
2. *Führe eine Bisektion von $G[U]$ durch. Dabei erhält man die Cluster X und Y.*
3. *Lösche das Cluster U aus C. Füge X und Y in C ein. Erhöhe c um 1.*
4. *Wenn gilt $k \neq c$: wähle aus C das Cluster W,welches als nächstes geschnitten werden soll, setze $U := W$ und gehe zu Schritt 2.*
5. *Gib C zurück.*

Die Clusterung C besitzt dann genau k Cluster. Schritt 2 wird anhand einer meist lokalen Kostenfunktion durchgeführt. Üblich sind zum Beispiel gewichtete Schnitte. Die Wahl des nächsten zu schneidenden Clusters in Schritt 4 erfordert eine Bewertung der Cluster.

Eine andere Interpretation der Operationen Linking und Splitting ist folgende: Die Durchführung eines Linkingschritts wird als eine Kantenkontraktion aufgefasst. Dabei werden die Knoten des Graphen G als die Cluster betrachtet. Bei der Kontraktion einer Kante $e = \{u,v\}$ wird die Kante e entfernt und die Knoten u und v werden miteinander identifiziert. Nun werden ausgehend vom Graph G nacheinander immer wieder Kantenkontraktionen durchgeführt. Sei k wieder die gewünschte Anzahl an Clustern. Gilt für den aus $G = (V,E)$ durch wiederholte Kantenkontraktionen entstandene Graph $G' = (V',E')$: $|V'| = k$, so stellen die Knoten von G' die gesuchten Cluster dar. Es ist also notwendig, sich zu merken, aus welchen Knoten von G die Knoten von G' entstanden sind. Dieses Verfahren entspricht einem Linkingalgorithmus beginnend mit der Clusterung, in der jeder Knoten von G in einem eigenen Cluster liegt.

Die Splitting Operation hingegen entspricht dem Schneiden einer Komponente des Graphen. Die Komponenten des Graphen werden dabei mit den Clustern indentifiziert.

2.4.2 Gewichtete Schnitte

Zur Durchführung eines Splittingschritts ist eine Kostenfunktion notwendig, die festlegt, welche Kanten aus dem Graph zu schneiden sind, damit dieser auf möglichst günstige

Art in eine bestimmte Anzahl Cluster zerfällt. Eine Variante dafür sind gewichtete Schnitte. Als Beispiel soll hier der Ratio Cut (RCut) nach Hagen und Kahng [11] dienen.

Definition 10 *Sei $G = (V,E)$ ein Graph und $C = \{C_1, \ldots, C_k\}$ eine Clusterung von G. Es wird definiert:*

$$\mathit{RCut}(G,C) := \mathit{Cut}(G,C)$$

und

$$\mathit{Size}_R(\mathit{RCut}(G,C)) := \frac{1}{2} \sum_{X \in C} \frac{|\mathit{Cut}(G, \{X, V \setminus X\}|}{|X|}$$

Man betrachtet also für je ein Cluster den 1-Schnitt, der entsteht, wenn man dieses Cluster vom Rest des Graphen trennt. Dessen Größe (size) wird dann durch die Mächtigkeit des Clusters geteilt. Dann wird der entstandene Wert über alle Cluster summiert. Erwähnenswert ist auch der Normalized Cut (NCut) nach Shi und Malik [10], welcher, anders als der Ratio Cut, mit der Anzahl der Kanten des vom entsprechenden Cluster C_i induzierten Untergraphen wichtet, und der in Abschnitt 2.5 eingeführte Conductance Cut (CCut).

Eine Verwendung von gewichteten Schnitten für Shifting-Algorithmen besteht darin, ausgehend von einer Startclusterung zu berechnen, inwieweit der Austausch zweier Knoten aus verschiedenen Clustern die Größe des durch die Clusterung induzierten Schnitts verändert. Selbiges gilt für das Verschieben eines Knotens in ein anderes Cluster und die dadurch veränderte Schnittgröße. Eine Variante zur Graphenbisektion nur unter Verwendung des Austauschs zweier Knoten ist der Kernighan-Lin-Algorithmus [14].

2.4.3 Kanten entfernen

Verwandt mit dem Zerlegen eines Graphen durch Schnitte ist das Zerlegen durch wiederholtes Entfernen von Kanten. Diese Methode ist aber nicht mit dem Zerlegen mittels Schnitten identisch. Auch hier werden die Komponenten des Graphen mit den Clustern identifiziert. Allerdings muss durch die Entfernung einer Kante noch kein Schnitt entstehen. Außerdem ist es möglich und wahrscheinlich, dass auch Kanten innerhalb von späteren Komponenten entfernt werden, bevor ein Schnitt entsteht. Zum Ermitteln, welche Kante zufällig entfernt werden soll, ist ein entsprechender Index notwendig. Der in dieser Arbeit untersuchte Clusteralgorithmus ist ein solcher Kantenentfernungs-Algorithmus der den Index Kanten-Betweenness verwendet.

2.4.4 Spektralmethoden

Eine weitere Möglichkeit einen Graphen in Cluster zu zerlegen sind Spektral-Methoden. Dabei wird im Fall einer Bisektion eine Zielfunktion als ein quadratisches Optimierungsproblem

$$s^T Ms \to \min$$

geschrieben. Der Vektor s stellt dabei einen Indikatorvektor dar, welcher die Zugehörigkeit eines Knotens v zum Cluster C_j angibt und dessen Betrag gewisse Schranken nicht übersteigen kann. Für s gelten je nach Zielfunktion weitere Bedingungen. M ist eine symmetische und reelle Matrix und ebenfalls von der zugrunde liegenden Zielfunktion abhängig. Die quadratische Form $s^T Ms$ kann dann unter Verwendung des Ritz-Reileigh-Theorems unter Verwendung der Eigenwerte von M optimiert werden. Dabei werden Teile der Anforderungen an s fallen gelassen. Diese müssen in einem Nachverfahren wieder hergestellt werden. Deswegen handelt es sich bei der Lösung aus der Spektralmethode nur um eine Approximation der Lösung der Optimierungsaufgabe. Für den Fall einer k-Sektion mit $k > 2$ geht das quadratische Optimierungsproblem in ein Spuroptimierungsproblem

$$\mathrm{tr}(S^T MS) \to \min$$

über, wobei S eine Indikatormatrix darstellt. Weitere Informationen zu Spektralmethoden sind in [16], [17], Hagen/Kahng und [13] zu finden.

2.5 Bewertung der Clusterung eines Graphen

Wie in Abschnitt 2.1 definiert wurde, ist eine Clusterung eines Graphen eine Partition der Knotenmenge. Dabei ist nicht bekannt, ob diese im Sinne des zugrunde liegenden Problems gut oder schlecht ist. Es ist möglich, einen Graphen genau so in die Ebene zu zeichnen, dass die Knoten, die im selben Cluster liegen, auch räumlich eng beieinander liegen und die einzelnen Cluster räumlich deutlich getrennt sind. Dadurch ist es möglich, einen Eindruck zu gewinnen, wie gut die Clusterung des Graphen die Anforderungen an eine gute Clusterung aus Abschnitt 2.3 erfüllt. Da verschiedene Varianten, den Graph in die Ebene zu zeichnen, zum Beispiel eine unterschiedliche Wahl von Koordinaten für die Knoten, auch einen anderen Eindruck erzeugen, ist ein solcher Eindruck recht subjektiv. Gesucht ist also eine Funktion, die einer gegebenen Clusterung C eines Graphen G einen Wert zuordnet, der eine Beurteilung ihrer Güte ermöglicht. Zu diesem Zweck werden nach [1] Clustergüteindizes der folgenden Form eingeführt:

$$\mathrm{index}(G,C) = \frac{f(G,C) + g(G,C)}{\max\{f(G,C') + g(G,C') : C' \in \mathscr{C}(G)\}}, \quad \exists C' : f(C') + g(C') \neq 0, \quad (2.1)$$

$$f, g, index : \mathscr{C}(G) \to \mathbb{R}^+$$

wobei f ein Maß für die Dichte innerhalb der Cluster ist (inter cluster density) und g ein Maß dafür, dass der Graph zwischen den Clustern nicht dicht ist (inter cluster sparsity). Mit $\mathscr{C}(G)$ ist die Menge aller Clusterungen von G bezeichnet. Der Nenner $\max\{f(G,C') + g(G,C')\}$ steht für den maximalen Wert, den die Zählerfunktion für eine Clusterung erreichen kann und stellt somit eine Normierung dar. Für einige der hier vorgestellten Clustergüteindizes ist dieser Wert trivial und bekannt. Für andere ist die Berechnung NP-Vollständig bezüglich der Zeit und es wird in der Praxis auf die Normierung verzichtet.

Die Güteindizes sind so konstruiert, dass einer guten Clusterung des Graphen ein hoher Wert für den Index zugeordnet wird. Allerdings steht der maximale Wert 1 oft für eine triviale Clusterung des Graphen, welche für die praktische Anwendung nicht sinnvoll ist. Weiterführende Informationen zu Clusterindizes sind in [1] zu finden.

2.5.1 Coverage

In Abschnitt 2.1 wurden Intra-Cluster-Kanten und Inter-Cluster-Kanten definiert. Nun ist es für eine gute Clusterung sicherlich sinnvoll, dass möglichst viele Kanten Intra-Cluster-Kanten sind. Der Index Coverage bewertet, wie hoch der Anteil der Inter-Cluster-Kanten an der Kantenmenge des gegebenen Graphen ist. Gesetzt wird:

$$f(G,C) = |E(C)|, \text{ und}$$
$$g(G,C) \equiv 0.$$

Nun ist der maximale Wert für f genau dann erreicht, wenn gilt: $E(C) = E$. Das ist zum Beispiel der Fall, wenn die Clusterstruktur der Zusammenhangsstruktur des Graphen folgt. Das heißt, f ist maximal, wenn jedes Cluster genau eine Zusammenhangskomponente des Graphen repräsentiert. Allgemeiner gilt $E(C) = E$, wenn jedes Cluster eine oder mehrere Zusammenhangskomponenten des Graphen komplett enthält. Trivial ist $E(C) = E$ erfüllt, wenn es nur ein einziges Cluster gibt, dass dann der Definition einer Clusterung eines Graphen nach alle Knoten und somit auch alle Kanten des Graphen enthält. Nun ist Coverage also folgendermaßen definiert [1]:

Definition 11 (Coverage) *Gegeben sei ein Graph G und eine Clusterung C des Graphen G.*

$$\textit{Coverage}(G,C) := \frac{|E(C)|}{|E|} \tag{2.2}$$

Die Definition von Coverage lässt sich auf gewichtete Graphen verallgemeinern. Coverage wird im Rahmen dieser Arbeit auch mit *cov* abgekürzt.

2.5.2 Performance

Im Abschnitt 2.5.1 wurden die Intra-Cluster-Kanten zur Bewertung der Dichte des Graphen innerhalb der Cluster herangezogen. Auch für Performance soll gelten:

$$f(G,C) := E(C).$$

Zusätzlich wird für den Index Performance eine Funktion g eingeführt, um zu bewerten, ob der Graph zwischen den Clustern nicht dicht ist (inter cluster sparsity). Ein Zählen der Inter-Cluster-Kanten ist dafür nicht sinnvoll, da deren Anzahl $|E| - |E(C)|$ beträgt und somit keine neue Information entsteht. Stattdessen wird für Performance g gleich der Anzahl der Inter-Cluster-Kanten gesetzt, die im vollständigen Graph K_n auf derselben Knotenmenge konstruiert wie unser betrachteter Graph $G = (V,E),\ |V| = n$ und die selbe Clusterung C zugrundegelegt, existieren würden, für G mit der Clusterung C aber nicht existieren. Die Sparsity-Funktion g hat dann folgende Form [1]:

$$g(G,C) := \binom{n}{2} - \sum_{i=1}^{k} \binom{|C_i|}{2} - |\overline{E(C)}|,\ k = |C|$$

Dabei ist $\binom{n}{2}$ die Kantenzahl des vollständigen Graphen K_n. Da wir am Anfang alle Kanten des K_n angesetzt haben, müssen wir auch die Cluster als vollständige Graphen betrachten. $\binom{|C_i|}{2}$ ist also Anzahl der Intra-Cluster-Kanten im Cluster C_i für den vollständigen Graph K_n mit der Clusterung C. Damit ist $\sum_{i=1}^{k} \binom{|C_i|}{2}$ die Anzahl der Intra-Cluster-Kanten. Wird diese von $\binom{n}{2}$ abgezogen, erhalten wir die Anzahl der Inter-Cluster-Kanten (für den Graphen K_n mit der Clusterung C). Gezählt werden soll aber nur der Anteil dieser Kanten, der in G nicht existiert. Somit wird am Schluss die Menge $|\overline{E(C)}|$ der in G existierenden Inter-Cluster-Kanten abgezogen.

Für Performance ist die Berechnung des Maximums $\max_{C \in \mathscr{C}}(f(G,C) + g(G,C))$ NP-vollständig bezüglich der Zeit. Deswegen wird dieser Wert in einigen Quellen mit der Kantenzahl $\binom{n}{2}$ des vollständigen Graphen K_n angenähert [1]. Sollte der Graph nur eine geringe Kantendichte besitzen, wird der Wert für Performance in diesem Fall aber für steigende n schnell klein. Das erschwert die Interpretation und kann auch numerische Probleme bereiten. Genau dieser Fall trifft aber auf viele in der Natur vorkommenden Netzwerke zu. Deswegen wird in dieser Arbeit auf eine Normierung von Performance verzichtet und der Index wird wie folgt definiert:

Definition 12 (Performance) *Sei $G = (V,E)$ ein Graph, sei $|V| = n$, sei C eine Clusterung von G und $|C| = k$. Dann wird definiert:*

$$\textit{performance}(G,C) := |E(C)| + \left(\binom{n}{2} - \sum_{i=1}^{k} \binom{|C_i|}{2} - |\overline{E(C)}| \right) \tag{2.3}$$

Performance wird im Rahmen dieser Arbeit auch mit *perf* abgekürzt.

2.5.3 Clusterindizes auf Basis gewichteter Schnitte

Eine weitere Variante, die Güte von Graphen-Clusterungen zu bewerten lässt sich über gewichtete Schnitte konstruieren. Dafür wird vereinbart:

Vereinbarung 2 *Sei $G = V,E$ ein Graph. Sei $X \subset V$ und $\overline{X} = V \setminus X$. Sei*

$$\mathit{WCut}(G, \{X, \overline{X}\}) := \mathit{Cut}(G, \{X, \overline{X}\})$$

ein gewichteter Schnitt und size_W ($\mathit{WCut}(G, \{X, \overline{X}\})$) eine Funktion, die dem Schnitt eine reelle Zahl zuordnet, und dabei die Balanciertheit der Cluster mit berücksichtigt. Diese Funktion soll Größe des gewichteten Schnitts heißen.

Sei $G = V,E$ der betrachtete Graph und C eine Clusterung dieses Graphen. Für die Konstruktion für Clustergüteindizes mithilfe gewichteter Schnitte werden hier folgend der Konstruktionen mithilfe der Conductance aus [1] zwei Wege vorgeschlagen:

1. Die erste Variante bewertet nur die innere Struktur der Cluster. Man betrachtet als erstes die von den Clustern C_i der Clusterung C induzierten Untergraphen $G[C_i]$. Für diese berechnet man jeweils den gewichteten Schnitt mit minimaler Größe, also $\min\limits_{X \subset C_i} \mathrm{size}_W(\mathrm{WCut}(G[C_i], \{X, C_i \setminus X\}))$. Wenn dieser Wert klein ist, gibt es noch mindestens einen billigen Schnitt. Dieses Cluster kann also noch einmal in zwei neue Cluster geteilt werden. Zum Bilden des Clustergüteindizes wird dann noch einmal das Maximum der Minimalschnitte der Cluster über alle Cluster betrachtet. Es gilt also:

   $$f(G,C) := \max_{C_i \in C} \min_{X \subset C_i} \mathrm{size}_W(\mathrm{WCut}(G[C_i], \{X, C_i \setminus X\})),$$
   $$g(G,C) \equiv 0.$$

   Wenn man sich das Cluster merkt, welches den Wert für $f(G,C)$ erzeugt hat und den Schnitt kennt, der diesen Wert besitzt, ist dadurch auch ein Splitting-Schritt zur Verbesserung für die Clusterung C zu einer Clusterung C' gegeben. Das entsprechende Cluster ist entsprechend der Schnittseiten X und $C_i \setminus X$ dieses Schnitts in zwei neue Cluster aufzuteilen.

   Allerdings ist die Berechnung minimaler gewichteter Schnitte NP-vollständig. Die Laufzeit der Berechnung ist exponentiell. Diese Berechnung muss für jeden Graphen $G[C_i]$ durchgeführt werden. Aufgrund dessen wird in dieser Arbeit keine Bewertung erzeugter Clusterungen mit einem so konstruierten Index durchgeführt.

2. Die zweite Variante zur Konstruktion von Clustergüteindizes bewertet, wie gut die Cluster vom Rest des Graphen separiert sind. Dazu wird für jedes Cluster die Größe des gewichteten Schnitts $\mathrm{WCut}(G, \{C_i, V \setminus C_i\})$ berechnet, die sich ergibt, wenn man dieses Cluster aus dem Rest des Graphen schneidet. Ist ein solcher Schnitt groß, so ist das entsprechende Cluster schlecht von den anderen Clustern, gegebenenfalls auch nur von einem anderen Cluster, separiert. Um den Index zu konstruieren, wird nun das Maximum der erhaltenen Werte für alle Cluster berechnet. Dieses wird vom am schlechtesten separierten Cluster erzeugt. Da für die Clustergüteindizes gelten soll, dass einer besseren Clusterung ein höherer Wert zugeordnet wird, ist der erhaltene Wert vom maximal für einen 1-Schnitt erreichbaren Wert abzuziehen. Es wird definiert:

$$
\begin{aligned}
&f(G,C) \equiv 0,\\
&g(G,C) := \max_{\mathrm{WCut}(G,\{\tilde{C}_i, V\setminus\tilde{C}_i\})} \mathrm{size}_W(\mathrm{WCut}(G,\{\tilde{C}_i, V\setminus\tilde{C}_i\}))\\
&\qquad\qquad - \max_{C_i \in C} \mathrm{WCut}(G,\{C_i, V\setminus C_i\}).
\end{aligned}
$$

Dabei gilt $\tilde{C}_i \in \tilde{C}$, $\tilde{C} \in \mathscr{C}$ und $|\tilde{C}| = 2$. Im Gegensatz zum Finden minimaler gewichteter Schnitte ist das Berechnen der Größe eines bekannten Schnitts in polynomialer Zeit möglich. Für RCut und Conductance liegt die Zeit dafür bei $O(|V|+|E|)$ und für NCut bei $O(|E|)$. Diese Berechnungen sind für jedes Cluster durchzuführen. Welche gewichteten Schnitte für diese Konstruktionen wirklich geeignet sind, wurde nicht untersucht.

Conductance

Ein weiteres Beispiel für einen gewichteten Schnitt ist die Conductance. Gegeben sei der Graph $G = (V,E)$ und ein Schnitt $\mathrm{Cut}(G, \{X, \overline{X}\})$ durch den Graph. Benennen wir als erstes die Anzahl der Kanten, die zu einem Knoten der Menge $X \in V$ inzident sind [1]:

Definition 13 *Sei $G = (V,E)$ ein Graph, sei $X \in V$. Dann ist $a(G,X)$ folgendermaßen definiert:*

$$
a(G,X) = \sum_{\{u,v\}\in E:\ u\in X \vee v \in V} 1
$$

Bemerkung 1 *Der Wert $a(G,X)$ lässt sich auch folgendermaßen darstellen:*

$$
a(G,X) = E(X) + E(X,\overline{X})
$$

Für den gewichteten Schnitt Conductance (Cond) wird der Schnitt $\mathrm{Cut}(G, \{X, \overline{X}\})$ mit dem Minimum von $a(G,X)$ und $a(G, V \setminus X)$ normiert. Dann ist die Conductance eines

Schnitts folgendermaßen definiert [1]:

Definition 14 (Conductance) *Sei $G = (V,E)$ ein Graph und $X \in V$. Sei Cut$(G,\{X,\overline{X}\})$ ein Schnitt durch G. Dann ist die Conductance dieses Schnitts folgendermaßen definiert:*

$$\mathit{Cond}(\mathit{Cut}(G,\{X,\overline{X}\})) := \begin{cases} 1 & \text{wenn } X \in \{\emptyset, V\} \\ 0 & \text{wenn } X \notin \{\emptyset, V\}, |E(X,\overline{X})| = 0 \\ \frac{|E(X,\overline{X})|}{|E(X,\overline{X})| + \min\{|E(X)|, |E(\overline{X})|\}} & \text{sonst.} \end{cases}$$

Die ersten beiden Zeilen dienen der Vermeidung einer Division durch null. Wie bei Schnitten üblich, stellt ein geringer Wert für Conductance einen guten Schnitt dar. Allerdings stellt Conductance allein noch keinen Clustergüteindex dar. Jedoch lässt sich ein solcher über die nach Abschnitt 2.5.3 konstruieren. Zu beachten ist, dass in der in dieser Arbeit verwendeten Definition des Begriffs Clusterung ein leeres Cluster nicht erlaubt wurde. Somit ist der Schnitt cut$(G,\{\emptyset,V\})$ hier nicht definiert. Allerdings liefert dieser verbotene Schnitt mit 1 eine obere Schranke für die erreichbaren Werte für Conductance. Der Wert 1 wird auch erreicht, wenn $X = \{v\}$ oder $V \setminus X = \{v\}, v \in V$ gilt.

Inter-Cluster-Conductance

Für die Konstruktion des Index Inter-Cluster-Conductance verwenden wir den eben vorgestellten gewichteten Schnitt Conductance und die zweite Variante zur Konstruktion von Clustergüteindizes aus Abschnitt 2.5.3. Als erstes betrachten wir eine Clusterung C eines Graphen $G = (V,E)$. Sei C_i ein Cluster von C. Dann gilt:

$$\mathrm{Cond}(\mathrm{Cut}(G,\{C_i,\overline{C_i}\})) = \begin{cases} 1 & \text{wenn } C_i \in \{\emptyset, V\} \\ 0 & \text{wenn } C_i \notin \{\emptyset, V\}, |E(C_i,\overline{C_i})| = 0 \\ \frac{|E(C_i,\overline{C_i})|}{|E(C_i,\overline{C_i})| + \min\{|E(C_i)|, |E(\overline{C_i})|\}} & \text{sonst} \end{cases}$$

Führen wir nun die Konstruktion des Index aus, erhalten wir die sogenannte Inter-Cluster-Conductance (InterCC) [1]:

Definition 15 (Inter-Cluster-Conductance) *Sei $(G = V,E)$ ein Graph und weiterhin sei $C = \{C_1, C_2, \ldots, C_k\}$ eine Clusterung von G. Dann ist die Inter-Cluster-Conductance folgendermaßen definiert:*

$$\mathit{InterCC}(G,C) := \begin{cases} 1 & \text{if } C = \{V\} \\ 1 - \max\limits_{i \in \{1,\ldots,k\}} (\mathit{Cond}(\mathit{Cut}(G,\{C_i,\overline{C_i}\}))) & \text{otherwise.} \end{cases} \tag{2.4}$$

Wie in Abschnitt 2.5.3 erwähnt, führt ein gut separiertes Cluster zu einem niedrigen gewichteten Schnitt. Somit wird für die Inter-Cluster-Conductance das am schlechtesten separierte Cluster für die Bewertung herangezogen. Sollte ein Cluster nur einen einzigen Knoten enthalten, liefert es den maximalen Conductance-Wert von 1 und somit der Inter-Cluster-Conductance-Wert von 0.

2.6 Zufälliges Erzeugen geclusterter Graphen

Im Rahmen dieser Arbeit wurden zum Testen des Newman-Girvan-Algorithmus zufällig erzeugte geclusterte Graphen verwendet. Zum Erzeugen eines zufälligen geclusterten Graphen $G = (V,E)$ wurden folgende Schritte durchgeführt:

1. Wähle eine Knotenmenge V, eine Intra-Cluster-Kanten-Wahrscheinlichkeit p und eine Inter-Cluster-Kanten-Wahrscheinlichkeit q.
2. Partitioniere die Knotenmenge. Dies entspricht dem Wählen einer Clusterung $C = \{C_1, \ldots, C_k\}$.
3. $G' = (V,E')$ sei der vollständige Graph mit der Knotenmenge V. Mit Wahrscheinlichkeit p wird eine Kante $e \in E'$ in E übernommen, die bezüglich C eine Intra-Cluster-Kante ist.
4. Mit Wahrscheinlichkeit q wird eine Kante $e \in E'$ in E übernommen, die bezüglich C eine Inter-Cluster-Kante ist.

Eine entsprechende Wahl von p und q vorausgesetzt, soll also ein Graph entstehen, für den C eine gute Clusterung ist. Damit eine geclusterte Struktur entsteht, muss $p >> q$ gelten, also die Wahrscheinlichkeit für eine Intra-Cluster-Kante größer sein, als die Wahrscheinlichkeit für eine Inter-Cluster-Kante. Für den Fall $p = 1$ und $q = 0$ ist C bezüglich jeden vorgestellten Cluster-Güte-Index maximal (mit Ausnahme der Intra-Cluster-Conductancel, wenn eines der Cluster nur aus einem einzelnen Knoten besteht). Umso größer der Graph wird, umso weiter müssen p und q auseinanderliegen. Außerdem muss auch mit einer feiner werdenden Clusterstruktur q verkleinert werden. Dies liegt daran, dass in beiden Fällen die Anzahl der möglichen Inter-Cluster-Kanten schneller wächst als die der Intra-Cluster-Kanten. Für einen Graph mit 30 Knoten und 5 gleichgroßen Clustern erwies sich eine Wahl von $p = 0.8$ und $q = 0.05$ als ausreichend für eine geclusterte Struktur. Diese Bewertung erfolgte anhand der Inter-Cluster-Conductance und es wurde angenommen, dass eine gute Clusterung einen Wert größer 0.5 erreichen sollte. Außerdem wurde auch für Coverage ein Wert von 0.5 als Mindestwert für eine gute Clusterung angenommen. Dies entspricht dem, dass die Hälfte der Kanten von G bezüglich der Clusterung C Intra-Cluster-Kanten sind. Beide Annahmen sind rein subjektiv gewählt. Für $p = 0.8$ und $q = 0.2$ bei $|V| = 30$ wurde in keinem Test eine Struktur erzeugt, für die C unter den obigen Annahmen eine gute Clusterung darstellt.

3 Betweenness

3.1 Wege und kürzeste Wege in Graphen

Der später eingeführte Index Betweenness basiert auf dem Zählen kürzester Wege. Deswegen wird in diesem Abschnitt definiert, was ein Weg bzw. ein kürzester Weg ist. Außerdem werden weitere, später benutzte Definitionen gegeben.

Als erstes ist zu klären, was unter einem Weg in einem Graphen verstanden wird:

Definition 16 (Weg, Teilweg, Weg von s nach t) *Sei $G = (V,E)$ ein Graph. Ein Weg P in G ist eine alternierende endliche Folge $v_1e_1v_2e_2 \ldots e_{m-1}v_m,\ m \leq |E|$ mit $v_i \in V$ und $e_i \in E$, für die gilt:*

1. *Aufeinanderfolgende Glieder der Folge sind in G inzident*
2. $v_i \neq v_j\ \forall i\, \forall j$

Eine Teilfolge $Q \subseteq P$, die wieder ein Weg ist, heißt Teilweg von P, schreibe $Q \leq P$.
Ein Weg von s nach t, $s,t \in V$ verbindet die Knoten s und t und es gilt: $v_1 = s$ und $v_m = t$.

Die Definition lässt eine Folge bestehend aus nur einem Knoten zu. Dementsprechend ist ein einzelner Knoten v bereits ein Weg. Sie lässt auch den leeren Weg zu. Mit dem Verbieten mehrfach vorkommender Knoten sind auch mehrfach vorkommende Kanten verboten. Um spätere Definitionen zu vereinfachen, ist folgendes Lemma nützlich:

Lemma 1 *Sei $G = (V,E)$ ein Graph. Sei $P \neq v,\ v \in V$ ein Weg in G. Der Weg P ist, abgesehen von seiner Durchlaufrichtung, durch die Menge seiner Kanten gegeben.*

Beweis: Sei $E' = \{e_1, e_2, \ldots, e_{m-1}\}$ die Menge aller Kanten des Weges P. Mit jeder Kante $e_i = \{v_i, v_{i+1}\},\ e_i \in E'$ sind auch die Knoten v_i und v_{i+1} bekannt. Da es laut Definition des Weges keinen Knoten v_k auf dem Weg P gibt, der zu keiner Kante e_i des Weges inzident ist, sind mit der Kenntnis von E' und G auch alle Knoten $v_i \in P$ bekannt. Durch die Inzidenz von v_i und e_i und die Inzidenz von e_i und v_{i+1} in G ist auch deren Reihenfolge P bekannt. Nicht bekannt ist die Richtung des Weges. □

Umgekehrt erzeugt nicht jede Kantenteilmenge $E' \subseteq E$ einen Weg. Über die Definition des Weges kann der Zusammenhang eines Graphen eingeführt werden:

Definition 17 (Zusammenhang, Zusammenhangskomponente) *Sei $G = (V,E)$ ein Graph.*

Ein Graph heißt zusammenhängend, wenn gilt:

$$\forall s,t \in V \exists \textit{Weg } P \textit{ zwischen } s \textit{ und } t.$$

Sei $X \in V$ und $G[X]$ zusammenhängend. Sei $Y \in V$ eine Teilmenge der Knotenmenge, die durch Aufnahme weiterer Knoten $v \in V \setminus X$ entstanden ist. Ein bezüglich dieser Aufnahme weiterer Knoten in die Menge X maximaler zusammenhängender Untergraph $G[Y]$ heißt Zusammenhangskomponente, kurz Komponente von G.

Für die Definition eines kürzesten Weges ist zu klären, was man unter der Länge eines Weges versteht. Außerdem lässt sich darüber der Abstand zweier Knoten definieren:

Definition 18 (Länge eines Weges, Abstand zweier Knoten, kürzester Weg zwischen zwei Knoten) *Sei $G = (V,E)$ ein Graph, sei $P = v_1 e_1 v_2 e_2 ... e_{m-1} v_m$, $m \leq |E|$ ein Weg in G. Sei $E' = \{e_1, e_2, \ldots, e_{m-1}\}$. Die Länge $l(P)$ eines Weges P ist definiert durch:*

$$l(P) := |E'|$$

Der Abstand $d(u,v)$ zweier Knoten $u,v \in V$ ist definiert durch:

$$d(u,v) := \begin{cases} \min\limits_{P: v_1 = u \wedge v_m = v} l(P) & \textit{falls } \exists P : v_1 = u \wedge v_m = v \\ \infty & \textit{sonst} \end{cases}$$

Ein solcher Weg von u nach v mit minimaler Länge (d.h. mit Länge $d(u,v)$) heißt kürzester Weg von u nach v.

Für $d(v,v)$ gilt laut Definitionen:

1. $\exists P : v_1 = v,\ v_m = v$, und zwar den Weg $P = v$.
2. $d(v,v) = l(v) = |E'|$, E' ist die Menge der Kanten in P.

Da $E' = \{\}$ folgt $d(v,v) = 0$.

Zur Vereinfachung späterer Definitionen werden folgende Bezeichnungen eingeführt:

Definition 19 $(P(E'), P_{uv})$
Sei $G = (V,E)$ ein Graph.

- *$P(E')$ sei ein Weg, der durch die Kantenteilmenge $E' \subseteq E$ gegeben ist.*
- *$P_{uv},\ u,v \in V$ sei ein kürzester Weg von u nach v.*

Dabei wird zur Definition von $P(E')$ das Lemma 1 benutzt. P_{uv} muss nicht eindeutig sein.

Die später eingeführten Algorithmen zur Berechnung eines Betweennessrankings bedienen sich Varianten einer Breitensuche (siehe Abschnitt 3.3.3). Diese verwendet die Tatsache, dass jeder Teilweg eines kürzesten Weges wieder ein kürzester Weg ist. Aufgrund dieser Tatsache kann eine Zerlegung eines kürzesten Weges in Teilwege der Länge 1 durchgeführt werden. Zur Herleitung dieser Eigenschaft sind die folgenden Definitionen notwendig:

Definition 20 (Verkettung von Wegen) *Sei $G = (V,E)$ ein Graph und $P = v_1e_1v_2e_2\dots v_k$ und $Q = v_ke_kv_{k+1}e_{k+1}v_{k+2}e_{k+2}\dots v_m$, $v_i \in V, e_i \in E$ für $i \in \{1,\dots,m\}$ Wege in G. Dann ist die Verkettung $P \odot Q$ von P und Q definiert durch:*

$$P \odot Q := v_1e_1v_2e_2\dots v_ke_kv_{k+1}e_{k+1}v_{k+2}e_{k+2}\dots v_m$$

und für eine Folge $\{P^l\}_{l=1}^{L}$ von Wegen durch:

$$\bigodot_{l=1}^{L} P^l := P^1 \odot P^2 \odot \dots \odot P^L.$$

Dabei ist zu beachten, dass die Verkettung von Wegen ist nicht für beliebige Wege definiert ist. Des weiteren muss eine Verkettung von Wegen im Allgemeinen kein Weg sein.

Mithilfe der Definition der Verkettung von Wegen lässt sich unter Verwendung der Verkettungsoperation eine Zerlegung eines Weges in Teilwege definieren:

Definition 21 (Zerlegung eines Weges P)
Sei $G = (V,E)$ ein Graph. Sei P ein Weg in G. Eine Menge

$$\{P^i : \exists \text{ eine Verkettung} \bigodot P^i : \bigodot P^i = P \ \wedge \ P^i \text{ ist Weg in } G, \forall i\}$$

heißt Zerlegung von P.

Lemma 2 (Länge von Teilwegen) *Sei $G = (V,E)$ ein Graph und P ein Weg in G. Für P und eine Zerlegung von P in Teilwege P^i, $i \in \{1,\dots,L\}$ gilt:*

$$l(P) = \sum_{i=1}^{L} l(P^i)$$

Beweis: Sei $E_i \in E$ die Menge von Kanten, die den Teilweg P_i erzeugt. Sei $E' \in E$ die Menge von Kanten, die den Weg P erzeugt. Für eine Zerlegung von P in Teilwege P_i gilt

nun:

$$E' = \bigcup_{i=1}^{L} E_i$$

Des weiteren gilt laut Definition 18:

$$l(P_i) = |E_i|$$

Zusammen mit der Disjunktheit der Mengen E_i folgt aus diesen beiden Aussagen:

$$l(P) = |E'| = \sum_{i=1}^{L} |E_i| = \sum_{i=1}^{L} l(P_i)$$

und somit die Aussage des Lemmas. □

Speziell gilt für kürzeste Wege folgender Satz:

Satz 1 *Jeder Teilweg eines kürzesten Weges ist ein kürzester Weg.*

Beweis: Sei $G = (V,E)$ der betrachtete Graph, $s,t \in V$ und P_{st} ein kürzester Weg von s nach t. Sei weiter P_{uv} ein beliebiger Teilweg von P_{st}. Annahme: Sei P_{uv} kein kürzester Weg.

- Dann gibt es einen Weg P'_{uv}, der kürzer ist als P_{uv}.
- Damit wäre es möglich, einen Weg $P'_{st} = P_{su} \odot P'_{uv} \odot P_{vt}$ zu konstruieren.
- Laut Lemma 2 gilt aber $l(P_{st}) = l(P_{su}) + l(P_{uv}) + l(P_{vt})$ und $l(P'_{st}) = l(P_{su}) + l(P'_{uv}) + l(P_{vt})$.

Da $l(P'_{uv}) < l(P_{uv})$ gilt, folgt $l(P'_{st}) < l(P_{st})$. Damit wurde ein kürzerer Weg von s nach t gefunden als P_{st}. Aber laut Annahme ist P_{st} ein kürzester Weg von s nach t. Damit kann kein kürzerer Weg von u nach v existieren als der Weg P_{uv}, welcher somit kürzester Weg ist. □

Eine Verkettung kürzester Wege ist im allgemeinen kein kürzester Weg. Sie muss nicht einmal ein Weg sein. Mit Satz 1 und Lemma 2 können folgende Schlüsse gezogen werden, wenn P_{st} kürzester Weg von s nach t ist:

Aus $\bigodot_i P^i = P_{st},\ i \in \{1,\ldots,L\}$ folgt $l(P^i) = d(u_i, v_i)$, wenn u_i der erste Knoten im Weg P^i ist und v_i der letzte ist. Außerdem gilt:

$$l(P_{st}) = d(s,t) = \sum_{i=1}^{L} d(u_i, v_i)$$

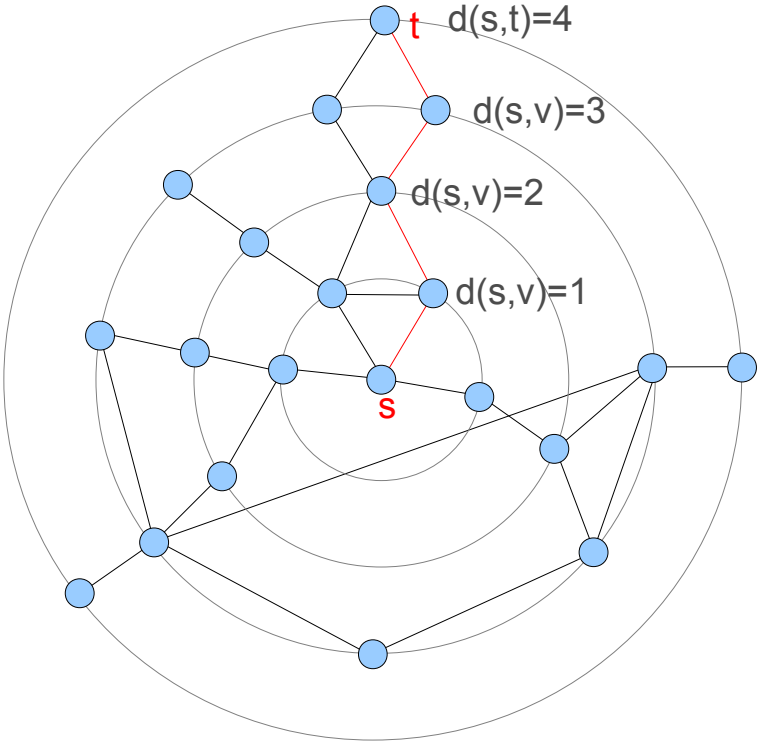


Abbildung 3.1: Von einem Startknoten s aus wurden alle Knoten mit aufsteigender Entfernung zu s auf Entfernungsradien gezeichnet. Dabei sind genau die Wege zwischen s und einem Knoten t kürzeste Wege, denen nur Knoten v von s aus mit je um 1 aufsteigender Distanz $d(s,v)$ bis $d(s,t)$ angehören.

Aus Satz 1 folgt dann das Lemma:

Lemma 3 *Sei $G = (V,E)$ ein Graph und $s, v_i \in V$, und $e_i \in E$, $i \in \{1,\ldots,k\}$. Wenn $P_{st} = se_1v_1e_2v_2\ldots v_k$ ein kürzester Weg in G von s nach t ist, dann gilt:*

$$d(s,v_i) = d(s,v_{i+1}) - 1 \; \forall i.$$

Somit ist es möglich, von einem Startknoten s aus alle kürzesten Wege zu jedem Knoten t zu finden, indem man die kürzesten Wege aus Wegen der Länge 1 aufbaut. Ein kürzester Weg von s nach v_i wird genau dann um die Kante $\{v_i, v_{i+1}\}$ und den Knoten v_{i+1} verlängert, wenn er zum Knoten v_i adjazent ist und sein Abstand $d(s,v_{i+1})$ zu s genau um 1 größer ist, als der Abstand $d(s,v_i)$ von s zu v_i.

Diese Konstruktion bildet die Grundlage für das Finden kürzester Wege mittels eines modifizierten Breitensuchalgorithmus. Die Zeichnung aus Abbildung 3.1 verdeutlicht

diesen Zusammenhang. Eine derartige Darstellung ist für jeden Graphen möglich.

3.2 Zentralitätsindex Betweenness

Zur Untersuchung sozialer Netzwerke wird oft ein Betweenness genannter Index verwendet, welcher einem Knoten eines auf einem solchen Netzwerk konstruierten Graphen einen positiven reellen Wert zur Bewertung seiner Lage im Graphen zuordnet. Grundlage dafür bildet der Informationsaustausch zwischen Akteuren des Netzwerks entlang kürzester Wege in dem auf dem Netzwerk konstruierten Graphen. Namentlich eingeführt wurde dieser Index 1977 von Freeman [8]. Allerdings wurde eine Variante der Betweenness unter der Bezeichnung *rush* in einem nie veröffentlichten technischen Report von Anthonisse [9] schon 1971 verwendet. Der Index Betweenness soll nun hergeleitet werden.

Betrachten wir ein soziales Netzwerk (siehe Abschnitt 2.3.2), repräsentiert durch den Graphen $G = (V,E)$, in dem ein Informationsaustausch zwischen den Personen stattfindet. Folgende Annahme wird getroffen:

Annahme 1 *Der Informationsaustausch zwischen zwei Personen repräsentiert durch die Knoten s und t , $s,t \in V$, findet immer entlang eines kürzesten Weges zwischen s und t statt.*

Als nächstes führen wir einen Index ein, der die Wichtigkeit eines Knotens für diese Informationsübertragung bewertet. Betrachten wir als erstes zwei Knoten s und t und den Informationsfluss zwischen diesen. Eine Information ausgehend vom Knoten s kann den Knoten t laut Annahme 1 nur über einen kürzesten Weg erreichen. Als Erstes benennen wir die Anzahl der kürzesten Wege ausgehend vom Knoten s hin zum Knoten t bezugnehmend auf Brandes [6] folgendermaßen:

Definition 22 *Sei $G = (V,E)$ ein Graph. Seien $s,t \in V$.*

$$\sigma_{st} := |\{P : P \text{ ist kürzester Weg von } s \text{ nach } t\}|.$$

Per Definition gibt es genau einen kürzesten Weg ausgehend von einem Knoten v zu sich selbst. Es gilt $d(v,v) = 0$. Es wird die Anzahl der kürzesten Wege benötigt, die von s nach t über einen dritten Knoten v laufen [6].

Definition 23 *Sei $G = (V,E)$ ein Graph. Seien $s,t,v \in V$. Es wird definiert:*

$$\sigma_{st}(v) := \begin{cases} |\{P : P \text{ ist kürzester Weg von } s \text{ nach } t \wedge v \leq P\}| & \text{wenn } v \neq s \wedge v \neq t \\ 0 & \text{sonst.} \end{cases}$$

Die Knoten s und t sollen laut obiger Definition nicht auf einem kürzesten Weg von s nach t liegen. Für das soziale Netzwerk aus Abschnitt 2.3.2 repräsentierten die Knoten des Graphen G die einzelnen Personen im Netzwerk. Es soll Annahme 1 gelten, das heißt, eine Information soll ausgehend von einer Person, repräsentiert durch den Knoten s, eine andere Person, repräsentiert durch den Knoten t, nur auf einem kürzesten Weg P_{st} bezüglich dem Graphen G erreichen können. Diese Annahme ist sicher nicht auf die Realität anzuwenden. Betrachten wir nun eine dritte Person, repräsentiert durch den Knoten v, und ihre Bedeutung für die Übertragung einer Information von der durch s repräsentierten Person ausgehend hin zur durch t repräsentierten Person. Liegt v auf allen kürzesten Wegen P_{st}, so ist v für die Informationsübertragung unerlässlich. Liegt v auf keinem dieser kürzesten Wege, so ist v für den Informationsaustausch von s nach t unbedeutend. Alles in allem gilt: Je größer der Anteil der kürzesten Wege, die über v laufen (von denen es $\sigma_{st}(v)$ viele gibt), an der Gesamtheit der kürzesten Wege von s nach t (wovon es σ_{st} viele gibt) ist, umso bedeutender ist v für die Übertragung einer Information von s nach t. Auf Basis dieser Erkenntnis lässt sich ein Index konstruieren [6]:

Definition 24 (Knoten-Paar-Betweenness) *Sei $G = (V,E)$ ein Graph. Sei $s,t,v \in V$. Dann ist die Knoten-Paar-Betweenness $\delta_{st}(v)$, folgendermaßen definiert:*

$$\delta_{st}(v) := \frac{\sigma_{st}(v)}{\sigma_{st}}.$$

Der Index $\delta_{st}(v)$ ist aber nur ein Maß für die Bedeutung eines Knotens bezüglich von genau zwei festgelegten anderen Knoten bei festgelegter Übertragungsrichtung. Die Bedeutung eines Knotens v für die Informationsübertragung in einem Netzwerk ist aber von der Bedeutung dieses Knotens für die Informationsübertragung zwischen allen Paaren von Knoten abhängig. Um alle Paare (s,t) zu berücksichtigen, wird für den Zentralitätsindex (Knoten-)Betweenness über alle Paare (s,t) summiert [6]:

Definition 25 (Knoten-Betweenness) *Sei $G = (V,E)$ ein Graph. Sei $s,t,v \in V$. Dann ist die Knoten-Betweenness $Z_B(v)$ auch Knoten-Mittler-Zentralität genannt, folgendermaßen definiert:*

$$Z_B(v) := \sum_{s,t \in V} \delta_{st}(v).$$

Je höher der Betweennesswert eines Knotens $v \in V$ ist, umso bedeutender ist dieser Knoten also für die Informationsübertragung im zugrundeliegenden Netzwerk und umso zentraler liegt der Knoten v bezüglich der Knoten-Betweenness im Graph. Da die Knoten s und t liegen laut Definition von $\sigma_{st}(v)$ nicht auf einem kürzesten Weg P_{st} liegen, gilt $\delta_{st}(s) = \delta_{st}(t) = 0$. Somit kann zur Berechnung von $Z_B(v)$ auch v aus der Summation

ausgeschlossen werden ($\sum_{s,t \in V} \delta_{st}(v) = \sum_{s,t \in V \setminus \{v\}} \delta_{st}(v)$).

In der obigen Definitionen für Knotenbetweenness Z_B wird für zwei Knoten s und t sowohl $\delta_{st}(v)$ als auch $\delta_{ts}(v)$ gezählt. Es gilt: $\delta_{st}(v) = \delta_{ts}(v)$. In anderen Definitionen wird ein Weg ohne Durchlaufrichtung definiert. Dann wird $\delta_{st}(v)$ nur einmal gezählt, da die Wege $v_1e_1v_2e_2 \dots e_{m-1}v_m$ und $v_me_{m-1} \dots e_2v_2e_1v_1$ miteinander identifiziert werden. Dabei halbiert sich der Wert bezüglich der Betweennesddefinition 25. Die hier benutzte Definition hat den Vorteil, dass sie zur (hier nicht vorgestellten) Betweenness-Definition für gerichtete Graphen passend ist.

Auf dieselbe Art wie für Personen und sie repräsentierende Knoten lässt sich auch eine Bewertung der Verbindung zwischen zwei Personen des sozialen Netzwerks aus Abschnitt 2.3.2 durchführen. Betrachten wir als erstes wieder zwei Knoten s und t mit Abstand $d(s,t) = k$. Des weiteren wird eine Kante $e = \{u,v\} \in E$ betrachtet, die eine Verbindung zwischen den durch u und v vertretenen Personen repräsentiert. Die durch u und v repräsentierten Personen können, müssen aber nicht die durch s und t repräsentierten Personen sein. Auch hier bildet die Informationsübertragung von s nach t unter Annahme 1 die Grundlage der Betrachtungen. Sicherlich ist es für die Wichtigkeit der durch e repräsentierten Verbindung von Bedeutung, auf wie vielen der kürzesten Wege von s nach t die Kante e liegt. Also wird analog zur Knoten-Betweenness definiert:

Definition 26 *Sei $G = (V,E)$ ein Graph. Sei $e = \{u,v\} \in E$. Dann ist*

$$\sigma_{st}(e) := |\{P : P \text{ ist kürzester Weg von } s \text{ nach } t \wedge uev \leq P\}|.$$

Allerdings ist hierbei nur berücksichtigt, auf wie vielen kürzesten Wegen von s nach t die Kante e liegt. Nehmen wir an, der Wert für $\sigma_{st}(e)$ sei hoch. Sollte die Kante e ausfallen, sind also viele dieser Informationswege unterbrochen. Es können aber immer noch Wege der Länge k existieren, die nicht über e laufen. Inwieweit die Kommunikation zwischen s und t durch den Ausfall von e gestört ist, hängt also auch von σ_{st} ab. Besser ist es also, den Anteil der kürzesten Wege von s nach t, die über die Kante e verlaufen, zu betrachten. Dazu wird eine Kanten-Paar-Betweenness eingeführt:

Definition 27 (Kanten-Paar-Betweenness) *Sei $G = (V,E)$ ein Graph. Sei $s,t \in V$, sei $e \in E$. Die Kanten-Paar-Betweenness $\delta_{st}(e)$ ist folgendermaßen definiert:*

$$\delta_{st}(e) := \frac{\sigma_{st}(e)}{\sigma_{st}}.$$

Um die Bedeutung der Kante e für die Informationsübertragung innerhalb eines Netzwerks zu ermitteln, wird nun $\delta_{st}(e)$ für alle Paare (s,t) ermittelt.

Der Zentralitätsindex Betweenness für eine Kante $e \in E$ ergibt sich nun folgendermaßen:

Definition 28 (Kanten-Betweenness) *Sei $G = (V,E)$ ein Graph. Die Kanten-Betweenness $Z_B(e)$ für eine Kante $e \in E$ ist folgendermaßen definiert:*

$$Z_B(e) := \sum_{s,t \in V} \delta_{st}(e)$$

Es bleibt immer noch die Wahl, ob man Wege als gerichtet annimmt und wie hier für s und t sowohl das Paar (s,t) als auch das Paar (t,s) betrachtet, oder im Fall ungerichteter Wege nur das Paar (s,t). Im zweiten Fall wird der Index Kantenbetweenness bezüglich zum ersten Fall halbiert.

Durch berechnen der Betweennesswerte für alle Knoten beziehungsweise für alle Kanten eines Graphen kann ein Betweennessranking (eine Bewertung und Sortierung anhand dieser Bewertung) für alle Knoten $v \in V$ beziehungsweise alle Kanten $e \in E$ durchgeführt werden.

3.3 Algorithmische Grundlagen

3.3.1 Repräsentation von Graphen im Computer

Für die algorithmische Behandlung graphentheoretischer Problemstellungen ist es notwendig, Graphen auch auf einem Rechner speichern zu können. Zur Speicherung von Graphen üblich sind vor allem folgende zwei Darstellungen. Die erste entspricht der Speicherung der Adjazenzmatrix, die andere sind sogenannte Adjazenzlisten.

Adjazenzmatrix

In der Adjazenzmatrix-Darstellung wird die Adjazenzmatrix des betrachteten Graphen als zweidimensionales Feld (Array) gespeichert.

Definition 29 *Sei G=(V,E) ein Graph. Sei $|V| = n$. Dann ist die Adjazenzmatrix $A = \{a_{ij}\}_{n \times n}$ folgendermaßen definiert:*

$$a_{ij} = \begin{cases} 1 & \textit{Knoten } i \textit{ ist zu Knoten } j \textit{ adjazent} \\ 0 & \textit{sonst.} \end{cases} \tag{3.1}$$

Vorteilhaft bei der Adjazenzmatrix-Darstellung ist, dass es schnell möglich ist, zu prüfen, ob ein Knoten u zu einem Knoten v adjazent ist. Allerdings wird ein Speicherplatz von

$|V|^2$ benötigt. Ihr Einsatz ist vor allem dann sinnvoll, wenn der betrachtete Graph dicht besetzt ist, d.h. es gilt $|E| \approx \binom{|V|}{2}$. Es gibt auch Algorithmen, für die eine Darstellung des Graphen in dieser Form Voraussetzung ist.

Adjazenzlisten

In der Adjazenzlisten-Darstellung eines Graphen existiert für jeden Knoten des Graphen eine Liste, in der die Knoten eingetragen sind, zu denen der entsprechende Knoten adjazent ist. Realisiert werden kann diese Darstellung über ein Feld, mit $|V|$ Listen. Dabei repräsentiert die k-te Stelle im Feld den Knoten k. Die k-te Liste enthält dann die Nummern der Knoten, die zum Knoten k adjazent sind. Diese Darstellung ermöglicht eine schnelle Auflistung aller Nachbarn eines Knotens.

3.3.2 FIFO-Warteschlange

Eine Warteschlange ist eine Datenstruktur, die in der Lage ist, eintreffende Objekte zu speichern, um sie auf Abruf in einer bestimmten Reihenfolge wieder zur Verfügung zu stellen. Eine FIFO-Warteschlange (First in first out) gibt die eingegebenen Objekte in der Reihenfolge wieder zurück, in der sie der Warteschlange übergeben wurden. Das heißt, ein Objekt, welches früher als ein anderes an die Warteschlange übergeben wurde, wird auch immer früher wieder zur Verfügung gestellt. Es wird für diese Arbeit angenommen, dass die verwendeten Warteschlangen beliebig viele Objekte speichern können.

3.3.3 Breitensuche

Bei der Breitensuche (Breadth First Search, kurz BFS) werden von einem Startknoten s aus alle durch einen Weg erreichbaren Knoten eines Graphen besucht. Dies geschieht auf folgende Weise:

Algorithmus 2
Sei $G = (V, E)$ ein Graph.

1. *Wähle einen Startknoten $s \in V$. Markiere s. Initialisiere eine FIFO-Warteschlange Q.*
2. *Setze: $v := s$.*
3. *Füge alle Knoten aus der Nachbarschaft von v in die Warteschlange Q ein, die noch nicht markiert sind.*
4. *Entnehme einen Knoten w aus der Warteschlange und markiere ihn.*
   Setze: $v := w$.
5. *Solange Q nicht leer ist, gehe zurück zu Schritt 3*

Wenn der letzte Knoten aus Q entfernt wird und keine neuen Knoten mehr eingefügt werden können, terminiert der Algorithmus. Existieren dann noch nicht markierte Knoten, ist der Graph nicht zusammenhängend. Es kann also mittels BFS ein Zusammenhangstest für einen Graphen durchgeführt werden. Durch diese Eigenschaft ist auch eine Ermittlung der Zusammenhangskomponenten von G möglich. Sollten nach Durchlauf des Algorithmus 2 noch nicht markierte Knoten existieren, so wird der Algorithmus für einen ebendieser Knoten als Startknoten neu gestartet. Es werden dann alle in einem Durchlauf markierten Knoten in eine Liste eingeordnet. Die in einer dieser Listen zusammengefassten Knoten repräsentieren eine Zusammenhangskomponente des Graphen G. Der Algorithmus 2 wird also so oft wiederholt, wie es Zusammenhangskomponenten in G gibt.

Eine andere Anwendung des BFS-Algorithmus ist die Suche eines bestimmten Knotens v ausgehend von einem Startknoten s. Dabei lässt sich mithilfe von Lemma 3 und den daraus gezogenen Schlussfolgerungen in Abschnitt 3.1 ein kürzester Weg von s nach v ermitteln. Dann terminiert der Algorithmus jedoch schon, wenn dieser Knoten gefunden wurde. Wurden alle Knoten der Komponente, in der s liegt, durchsucht und v nicht gefunden, existiert kein Weg von s nach v. Es ist möglich, in einem Durchlauf, alle kürzesten Wege, beginnend vom Startknoten s zu jedem Knoten $v \in V$, zu ermitteln.

Des Weiteren kann man, anstelle der Markierung der Knoten eine Bewertung setzen. Durch die Eigenschaften der Warteschlange werden im BFS-Algorithmus jeweils erst nacheinander alle Knoten mit Abstand k zum Startknoten s besucht und erst wenn diese alle abgearbeitet sind, alle Knoten mit Abstand $k+1$. Dabei liegt k zwischen 0 und dem maximalen endlichen Abstand, den ein Knoten v zu s hat minus eins (Durchmesser der Komponente von G, in der s liegt) . Es kann allen Knoten des Graphen mittels BFS einen Abstand zum Startknoten s zugeordnet werden, indem man im Schritt 3, bei der Durchsuchung der Nachbarschaft von v, jedem Knoten u, dem noch keine Entfernung zugeordnet wurde, eine Entfernung $d(s,u) = d(s,v)+1$ zuordnet. Dazu ist im Schritt 1 $d(s,s) := 0$ zu setzen. Sollte ein Knoten x mit Durchlauf des Algorithmus 2 für Knoten s nicht erreicht werden, gilt $d(s,x) = \infty$.

Zum Zählen kürzester Wege zwischen dem Startknoten s und jedem Knoten $v \in V$ wird die Menge der Vorgänger eines Knotens bezüglich s benötigt:

Definition 30 *Sei $G = (V,E)$ ein Graph. Seien $u,v,s \in V$. Dann ist die Menge der Vorgänger eines Knotens v bezüglich eines Knotens s folgendermaßen definiert:*

$$\mathscr{P}_s(v) = \{u \in V : \ \{u,v\} \in E, d(s,v) = d(s,u)+1\}$$

In Abbildung 3.1 auf Seite 23 ist die Menge der Vorgänger eines Knotens v bezüglich eines Knotens s genau die Menge der Nachbarn von v, die auf dem nächstniedrigeren Abstandsradius liegen.

Weiter gilt nach [6] folgendes Lemma, welches in Abbildung 3.2 verdeutlicht ist:

Lemma 4 *Sei $G = (V,E)$ ein Graph. Seien $s, v, u \in V$. Dann gilt:*

$$\sigma_{sv} = \sum_{u \in \mathscr{P}_s(v)} \sigma_{su}$$

Diese Eigenschaft ermöglicht das Zählen aller kürzesten Wege vom Startknoten s aus zu allen anderen Knoten $v \in V$ mittels BFS und lässt sich unter Zuhilfenahme von Lemma 3 beweisen.

Im Schritt 1 des Algorithmus 3 aus Abschnitt 3.5 findet gleichzeitig das Zuordnen des Abstands $d(s,v)$ wie auch das Zählen der kürzesten Wege auf die hier beschriebene Weise statt. Für den Schritt 2 dieses Algorithmus wird dann eine Art Rückwärts-Breitensuche verwendet. Dabei werden von den Knoten mit dem größten Abstand zum Startknoten s aus alle Knoten (und somit Kanten) der Komponente, in der s liegt, mit kleiner werdendem Abstand bis zum Erreichen von s durchsucht. Dafür müssen diese Abstände bereits bekannt sein.

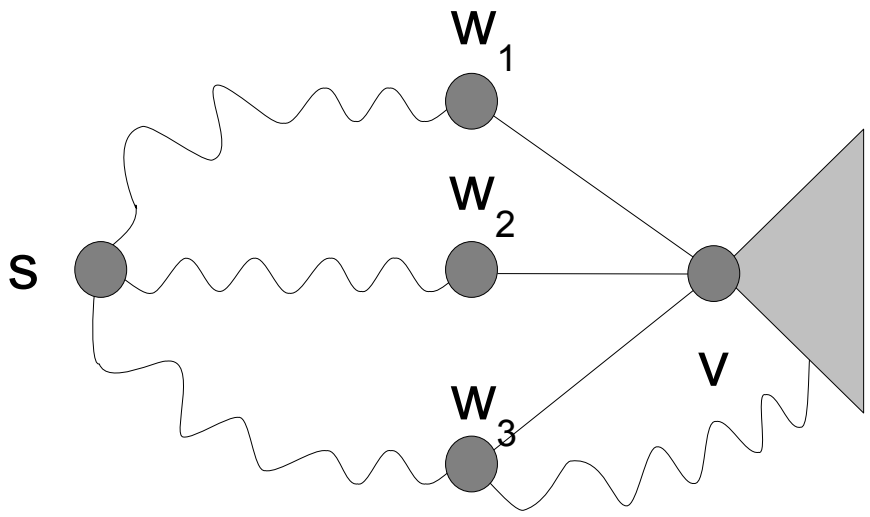


Abbildung 3.2: Beispiel zu Lemma 4. Es soll $d(s,v) = d(s,w_i) + 1,\ i \in \{1,2,3\}$ gelten. Außerdem sind w_1, w_2 und w_3 zu v adjazent. Damit gilt nach Lemma 4: $\sigma_{sv} = \sigma_{sw_1} + \sigma_{sw_2} + \sigma_{sw_3}$.

Eine Breitensuche benötigt eine Zeit von $O(n+m)$, da jeder Knoten genau einmal in Q

aufgenommen wird und im Rahmen der Ermittlung der Nachbarn jede Kante genau einmal betrachtet werden muss. Für große Graphen wird angenommen, dass die Kantenmenge eine wesentlich größere Mächtigkeit besitzt als die Knotenmenge. Deshalb wird die Laufzeit des Breitensuchalgorithmus für gewöhnlich mit $O(m)$ angegeben. Da bei der Breitensuche für jeden Knoten die Nachbarschaft ermittelt wird, empfiehlt sich eine Darstellung des Graphen über Adjazenzlisten. Für dicht besetzte Graphen ($|E| \approx \binom{|V|}{2}$) kann aber auch die Darstellung über die Adjazenzmatrix verwendet werden. Weitere Informationen zur Breitensuche sind in [2] zu finden.

3.4 Berechnung eines Betweennessrankings

3.4.1 Klassische Berechnung eines Betweennessrankings

In Abschnitt 3.2 wurde der Zentralitätsindex Betweenness für Knoten beziehungsweise Kanten eines Graphen $G = (V,E)$ eingeführt. Zur Berechnung wird üblicherweise für ein Knotenpaar (s,t) die Anzahl σ_{st} der kürzesten Wege von s nach t sowie der Wert $\sigma_{st}(v)$ für alle Knoten $v \in V$ beziehungsweise $\sigma_{st}(e)$ für alle Kanten $e \in E$ berechnet. Damit erhält man die Paar-Betweenness-Werte $\delta_{st}(v)$ beziehungsweise $\delta_{st}(e)$. Diese Berechnung ist mithilfe einer modifizierten Breitensuche in linearer Zeit O(n+m) beziehungsweise O(m) für $n << m$ möglich. Zur Berechnung eines Betweennessrankings müssen weiterhin alle Knotenpaare $(s,t) \in V \times V$ betrachtet werden. Das führt auf eine Gesamtkomplexität von $O(n^2(n+m))$ beziehungsweise $O(n^2m)$ für $n << m$.

3.4.2 Rekursionsformel für akkumulierte Paar-Betweenness nach Newman und Girvan und unabhängig nach Brandes

An der Stelle, an der in der klassischen Berechnung eines Betweennessrankings alle kürzesten Wege von s nach t berechnet werden, ist es mit geringer Modifikation des Breitensuchalgorithmus auch möglich, die Anzahl der kürzesten Wege vom Knoten s aus zu **allen** Knoten t unter Verwendung von Lemma 4 zu ermitteln. Ebenfalls kann an dieser Stelle auf die in Abschnitt 3.3.3 erklärte Art der Abstand von s zu jedem dieser Knoten t ermittelt werden. Eine entscheidende Beschleunigung wäre erreichbar, wenn nun auch die Anzahl der kürzesten Wege von Knoten s zu Knoten t, die über einen Knoten v laufen ($\sigma_{st}(v)$) für die Berechnung der Knotenbetweenness beziehungsweise die über eine Kante e laufen $(\sigma_{st}(e))$ im Fall der Berechnung der Kantenbetweenness, auf ähnliche Art ermittelbar wären. Diese Information wäre auch mit der Paar-Betweenness gegeben. Ein derartiger Algorithmus wurde sowohl von Newman [5] als auch unabhängig von Brandes [6] 2001 für Knoten-Betweenness vorgestellt. Eine Umsetzung für Kanten-Betweenness wurde 2003 von Newman und Girvan vorgestellt und 2004 veröffentlicht [7].

Als erstes wird die akkumulierte Paar-Betweenness benötigt:

Definition 31 *Sei $G = (V,E)$ der betrachtete Graph. Seien $s, v \in V$ und $e \in E$. Dann ist die akkumulierte Paar-Betweenness:*

$$\delta_{s\bullet}(v) = \sum_{t \in V} \delta_{st}(v)$$

für die Berechnung der Knoten-Betweenness ([6]) und

$$\delta_{s\bullet}(e) = \sum_{t \in V} \delta_{st}(e)$$

für die Berechnung der Kanten-Betweenness.

Die Rekursionsformel nach Newman beziehungsweise Brandes ermöglicht die Berechnung der akkumulierten Paar-Betweenness-Werte für einen Startknoten s und für alle Knoten $v \in V$ im Falle der Berechnung eines Knotenbetweenness-Rankings beziehungsweise für alle Kanten $e \in E$ im Falle der Berechnung eines Kantenbetweenness-Rankings.

Es gilt für die Knoten-Betweenness $Z_B(v)$:

$$\begin{aligned} Z_B(v) &= \sum_{s,t \in V} \delta_{st}(v) \\ &= \sum_{s \in V} \sum_{t \in V} \delta_{st}(v) \\ &= \sum_{s \in V} \delta_{s\bullet}(v) \end{aligned}$$

und für die Kanten-Betweenness $Z_B(e)$:

$$\begin{aligned} Z_B(e) &= \sum_{s,t \in V} \delta_{st}(e) \\ &= \sum_{s \in V} \sum_{t \in V} \delta_{st}(e) \\ &= \sum_{s \in V} \delta_{s\bullet}(e). \end{aligned}$$

Mit der akkumulierten Knoten-Paar-Betweenness $\delta_{s\bullet}(v)$ für alle Knoten $s \in V$ ist also auch die Knoten-Betweenness $Z_B(v)$ bekannt. Selbiges gilt für die akkumulierte Kanten-Paar-Betweenness $\delta_{s\bullet}(e)$ und die Kanten-Betweenness $Z_B(e)$.

3.4.3 Rekursionsformel für akkumulierte Knoten-Paar-Betweenness

Zur Veranschaulichung des Sachverhalts wird die Rekursionsformel zuerst für einen Baum aufgestellt:

Betrachtungen für einen Baum

Für einen Baum gilt, dass es exakt einen Weg von einem Knoten s zu einem Knoten t gibt. Dieser Weg ist damit zwangsläufig der kürzeste Weg. Ein Baum sei als zusammenhängender Graph mit genau dieser Eigenschaft definiert. Ein Knoten v kann also entweder auf diesem Weg liegen oder nicht und es gilt:

$$\sigma_{st}(v) = \begin{cases} 1 & \text{wenn } v \text{ auf einem kürzesten Weg zwischen } s \text{ und } t \text{ liegt} \\ 0 & \text{sonst.} \end{cases} \tag{3.2}$$

Als nächstes wird das Bellman Kriterium benötigt [6]:

Lemma 5 (Bellman Kriterium) *Sei $G = (V,E)$ ein Graph. Sei $s,t,v \in V$. Der Knoten v liegt genau dann auf einem kürzesten Weg zwischen s und t, wenn gilt: $d(s,t) = d(s,v) + d(v,t)$.*

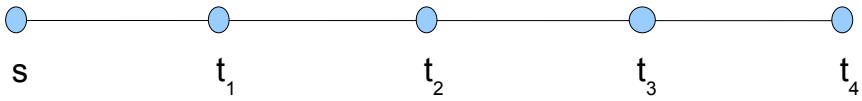
Beweis: Gegeben seien die Bedingungen aus Lemma 5. Per Definition ist $d(s,t)$ die Länge eines kürzesten Weges P_{st} zwischen s und t. Weiter ist $d(s,v) + d(v,t)$ die Länge eines kürzesten Wegen von s nach v plus die Länge eines kürzesten Weges von v nach t. Damit ist $d(s,v) + d(v,t)$ die minimale Länge eines Weges von s nach t der über v führt.

1. Nehmen wir an, es gibt einen kürzeren Weg von s nach t, der nicht über v führt. Dann gilt aber $d(s,t) < d(s,v) + d(v,t)$.
2. Nehmen wir stattdessen an, es gilt $d(s,t) > d(s,v) + d(v,t)$. Dann hätten wir aber einen Weg von s nach t über v gefunden, der kürzer ist als der Weg, der $d(s,t)$ erzeugt hat. Dieser ist aber per Definition minimal.

□

Das Lemma verwendet also die Tatsache, dass ein Teilweg eines kürzesten Weges stets wieder ein kürzester Weg ist. Setzt man das Bellmann-Kriterium in Beziehung (3.2) ein, erhält man:

$$\sigma_{st}(v) = \begin{cases} 1 & d(s,t) = d(s,v) + d(v,t) \\ 0 & \text{sonst.} \end{cases} \tag{3.3}$$



	v=s		v=t_1		v=t_2		v=t_3		v=t_4
d(s,v)	0		1		2		3		4
σ_{sv}	1		1		1		1		1
σ_{st}(v), d(s,t)> d(s,v)	0		1		1		1		-
σ_{st}(v), d(s,t)≤ d(s,v)	0		0		0		0		0
δ_{st}(v), d(s,t)> d(s,v)	0		1		1		1		-
δ_{st}(v), d(s,t)≤ d(s,v)	0		0		0		0		0
$\delta_{s\bullet}$(v)	0		3		2		1		0

Abbildung 3.3: Akkumulierte Knoten-Paar-Betweenness für einen Weg.

Da es im Baum nur einen kürzesten Weg P_{st} gibt, gilt: $\delta_{st}(v) = \sigma_{st}(v)$. Sollte der Graph ein Weg, also ein Baum mit $\forall v:\ \deg(v) \leq 2$, und s ein Blatt sein, dann gilt für die akkumulierte Knoten-Paar-Betweenness folgendes:

$$\delta_{s\bullet}(v) = \max_{t \in V}(d(s,t)) - d(s,v),\ v \neq s. \tag{3.4}$$

In Abbildung 3.3 wird diese Beziehung dargestellt. Mit jedem Schritt, den man dem Startknoten s näher kommt, erhöht sich $\delta_{s\bullet}(v)$ um eins. Im Falle der Knoten t_1 und t_2 geschieht dies zum Beispiel, weil für $t_1 = v$ auch t_2 ein zu berücksichtigender Knoten t mit $d(s,v) < d(s,t)$ ist, und somit $\delta_{st_2}(t_1) = 1$ mit in $\delta_{s\bullet}(t_1)$ eingeht. In $\delta_{s\bullet}(t_2)$ geht aber für t_2 der Wert $\delta_{st_2}(t_2) = 0$ ein.

In Abbildung 3.4 wird ersichtlich, dass für einen Baum folgendes Lemma gilt [6]:

Lemma 6 (Brandes) *Sei $G = (V,E)$ ein Baum. Seien $s,v \in V$ und $w \in V$ für alle w. Dann gilt:*

$$\delta_{s\bullet}(v) = \sum_{w:\ v \in \mathscr{P}_s(w)} (1 + \delta_{s\bullet}(w)). \tag{3.5}$$

Der folgende Beweis orientiert sich an dem Beweis von Brandes aus [6].

Beweis: Seien die Bedingungen von Lemma 6 erfüllt. Ein Knoten $v \in V$ liegt aufgrund der Baumstruktur für ein Knotenpaar $s,t \in V$ entweder auf allen kürzesten Wegen zwischen s und t oder auf keinem. Somit gilt wieder:

$$\delta_{st} = \begin{cases} 1 & d(s,t) = d(s,v) + d(v,t) \\ 0 & \text{sonst.} \end{cases} \tag{3.6}$$

Im Gegensatz zu einem Weg-Graphen mit $\deg(s) = 1$ kann es im Baum für einen Knoten v mehrere Knoten w geben, für die $v \in \mathscr{P}_s(w)$ gilt. Der Knoten v liegt dann auf jedem kürzesten Weg zwischen s und einem Knoten $t \in V$, auf dem auch der Knoten w liegt. Allerdings liegt v auch auf genau einem kürzesten Weg für jeden derartigen Knoten w mehr, nämlich auf dem Weg P_{sw}. Auf anderen kürzesten Wegen kann v aber nicht liegen. Das geht hervor aus Lemma 5. Zu beachten ist die Tatsache, dass für jeden Knoten $t \in V$, für den sowohl $d(s,t) = d(s,v) + d(v,t)$ als auch $d(s,t) > d(s,v) + 1$ gilt, es genau einen Knoten w mit obigen Eigenschaften gibt, der auf P_{st} liegt. Es gilt also:

$$\begin{aligned} \delta_{s\bullet}(v) &= \sum_{t \in V} \delta_{st}(v) && (3.7) \\ &= \sum_{w:\ v \in \mathscr{P}_s(w)} (1 + \sum_{t \in V} \delta_{st}(w)) && (3.8) \\ &= \sum_{w:\ v \in \mathscr{P}_s(w)} (1 + \delta_{s\bullet}(w)). && (3.9) \end{aligned}$$

□

Rekursionsformel für beliebige Graphen

Für die akkumulierte Knoten-Paar-Betweenness gilt folgende Rekursion [6]:

Satz 2 (Brandes) *Sei $G = (V,E)$ ein Graph. Seien $s,t,v,w \in V$ und $s \neq v$.*

$$\delta_{s\bullet}(v) = \sum_{w:\ v \in \mathscr{P}_s(w)} \frac{\sigma_{sv}}{\sigma_{sw}} \cdot (1 + \delta_{s\bullet}(w)).$$

Es soll die Rekursionsformel für die akkumulierte Knoten-Betweenness hergeleitet werden. Die Herleitung orientiert sich dabei an dem Beweis von Brandes aus [6].

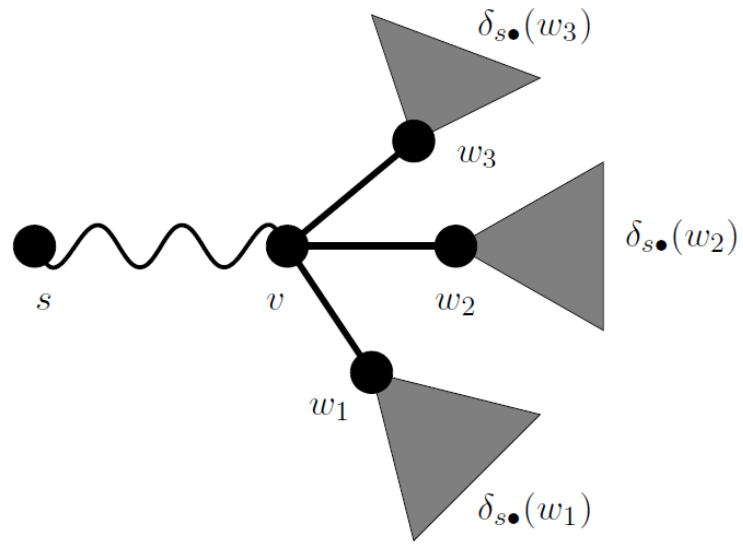


Abbildung 3.4: Akkumulierte Knoten-Paar-Betweenness für einen Baum. Ein kürzester Weg von s nach t, der über v läuft, startet beim Knoten s und läuft über den Knoten v und dann über die Kante $\{v, w_i\}$ zum Knoten w_i.

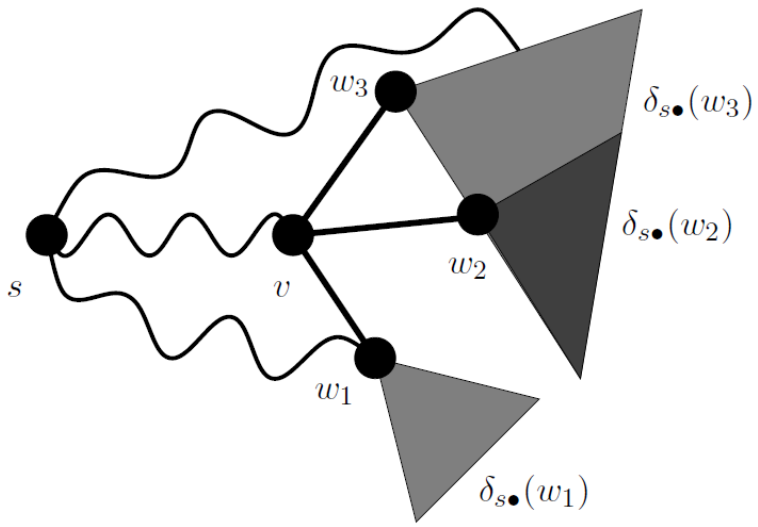


Abbildung 3.5: Allgemeiner Fall für die akkumulierte Knoten-Paar-Betweenness. Ein kürzester Weg von s nach t , der über v läuft, startet beim Knoten s und läuft über den Knoten v und dann über die Kante $\{v, w_i\}$ zum Knoten w_i. Es kann dabei noch kürzeste Wege von s zu einem Knoten t mit $d(s,t) > d(s,v)$ geben, die nicht über v laufen.

Beweis:
Es gilt folgende Eigenschaft für $\sigma_{st}(v)$:

$$\sigma_{st}(v) = \sigma_{sv} \cdot \sigma_{vt}. \tag{3.10}$$

Es wird eine weitere Form der Paar-Betweenness benötigt [6]:

Definition 32 *Sei $G = (V,E)$ ein Graph. Seien $s,t,v,w \in V$, sei $\{v,w\} \in E$. Wir definieren:*

$$\sigma_{st}(v,\{v,w\}) := |\{P : \ v\{v,w\}w \leq P \wedge v \in \mathscr{P}_s(w) \wedge P \textit{ ist kürzester Weg von } s \textit{ nach } t.\}|$$

und

$$\delta_{st}(v,\{v,w\}) := \frac{\sigma_{st}(v,\{v,w\})}{\sigma_{st}}.$$

Mit $\sigma_{st}(v,\{v,w\})$ werden also alle kürzesten Wege gezählt, die von s aus erst über den Knoten v laufen und dann über die Kante $\{v,w\}$ zum Knoten w. Damit lässt sich folgende Umformung durchführen [6]:

$$\begin{aligned}
\delta_{s\bullet}(v) &= \sum_{t \in V} \delta_{st}(v) && (3.11)\\
&= \sum_{t \in V} \sum_{w:\ v \in \mathscr{P}_s(w)} \delta_{st}(v,\{v,w\}) && (3.12)\\
&= \sum_{w:\ v \in \mathscr{P}_s(w)} \sum_{t \in V} \delta_{st}(v,\{v,w\}). && (3.13)
\end{aligned}$$

Es ist zu beachten, dass $\delta_{st}(v) > 0$ nur für genau die $t \in V$ gilt, für die v auch auf mindestens einem kürzesten Weg von s nach t liegt. Betrachten wir einen konkreten dieser kürzesten Wege (d.h. t und w sind fest). Nehmen wir weiter an, der Weg verläuft von s kommend über v und dann über die Kante $\{v,w\}$ nach w um von w zu t zu laufen. Es gibt für jeden derartigen kürzesten Weg genau eine solche Kante $\{v,w\}$ und damit auch einen festen Knoten w für jeden konkreten Weg. Das heißt, wenn in Zeile (3.12) der obigen Umformung über alle $w : \ v \in \mathscr{P}_s(w)$ summiert wird, zählt man tatsächlich jeden kürzesten Weg mit, der auch schon für $\delta_{st}(v)$ berücksichtigt wurde. Für alle t, für die schon $\delta_{st}(v) = 0$ galt, gilt wieder $\delta_{st}(v,\{v,w\}) = 0$.

Wie in Abbildung 3.5 ersichtlich, gilt für die Anzahl $\sigma_{st}(v,\{v,w\})$ solcher Wege folgende Beziehung:

$$\sigma_{st}(v,\{v,w\}) = \sigma_{sv} \cdot \sigma_{wt}. \tag{3.14}$$

Dies folgt auch aus einer einfachen kombinatorischen Betrachtung. Weiter ist aus Glei-

chung (3.10) bekannt [6]:

$$\sigma_{st}(w) = \sigma_{sw} \cdot \sigma_{wt}. \tag{3.15}$$

Umgestellt ergibt das:

$$\sigma_{wt} = \frac{\sigma_{st}(w)}{\sigma_{sw}}. \tag{3.16}$$

Mit den Gleichungen (3.14) und (3.16) ergibt sich dann [6]:

$$\delta_{st}(v,\{v,w\}) = \begin{cases} \frac{\sigma_{sv}}{\sigma_{sw}} & \text{wenn } t = w \\ \frac{\sigma_{sv} \cdot \sigma_{st}(w)}{\sigma_{st}} = \frac{\sigma_{sw}}{\sigma_{st}} & \text{wenn } t \neq w. \end{cases} \tag{3.17}$$

Eingesetzt in Gleichung (3.13) ergibt sich [6]:

$$\sum_{w:\ v \in \mathscr{P}_s(w)} \sum_{t \in V} \delta_{st}(v,\{v,w\}) = \sum_{w:\ v \in \mathscr{P}_s(w)} \left(\frac{\sigma_{sv}}{\sigma_{sw}} + \sum_{t \in V \setminus \{w\}} \frac{\sigma_{sv}}{\sigma_{sw}} \cdot \frac{\sigma_{st}(w)}{\sigma_{st}} \right) \tag{3.18}$$

$$= \sum_{w:\ v \in \mathscr{P}_s(w)} \frac{\sigma_{sv}}{\sigma_{sw}} \cdot (1 + \delta_{s\bullet}(w)). \tag{3.19}$$

□

In Zeile (3.18) wird w aus der Summation ausgeschlossen, da der Term durch Einsetzen von $\delta_{sw}(v,\{vw\}) = 1$ entstanden ist. In der darauffolgenden Zeile darf für $\delta_{st}(w)$ dann trotzdem wieder über alle t summiert werden, da für alle t mit $d(s,t) \leq d(s,v)$ gilt: $\delta_{st}(w) = 0$ und somit auch für w.

3.4.4 Rekursionsformel für akkumulierte Kanten-Paar-Betweenness

Betrachtungen für einen Weggraphen und ein Blatt als Startknoten

Nehmen wir vorerst an, der gegebene Graph $G = (V,E)$ sei ein Weg. Seien $s,t,u,v \in V$ und sei s ein Blatt. Sei $e = \{u,v\} \in E$. Dann liegt jede Kante abhängig von der Wahl von t entweder auf jedem kürzesten Weg von s nach t oder auf keinem:

$$\sigma_{st}(\{u,v\}) = \begin{cases} 1 & d(s,u) < d(s,t) \\ 0 & \text{sonst.} \end{cases} \tag{3.20}$$

Des weiteren gibt es vom Knoten s aus zu jedem Knoten t genau einen kürzesten Weg und es gilt:

$$\delta_{st}(\{u,v\}) = \sigma_{st}(\{u,v\}) \tag{3.21}$$

Betrachten wir nun $\delta_{s\bullet}(\{u,v\})$, können wir beobachten:

$$\delta_{s\bullet}(\{u,v\}) = \sum_{t \in V} \delta_{st}(\{u,v\}) \tag{3.22}$$

$$= \sum_{t \in V:\ d(s,u) < d(s,t)} 1. \tag{3.23}$$

Definieren wir die Menge der Vorgänger der Kante $\{v,w\}$ bezüglich dem Knoten s:

Definition 33 *Sei $G = (V,E)$ ein Graph. Seien $s,u,v,w \in V$. Gelte $d(s,v) < d(s,w)$. Dann ist die Menge der Vorgänger $P_s(\{v,w\})$ der Kante $\{v,w\} \in E$ bezüglich des Knotens s folgendermaßen definiert:*

$$\mathscr{P}_s(\{v,w\}) = \{\{u,v\} \in E : d(s,u) = d(s,v) - 1\}.$$

Sei weiter $f \in E$ eine Kante, für die gilt: $e \in \mathscr{P}_s(f)$. Dann gilt im Weggraphen: $\delta_{st}(e) = \delta_{st}(f)$ für alle t außer für $t = v$. Für $t = v$ gilt: $\delta_{st}(e) = 1$ aber $\delta_{st}(f) = 0$.

Daraus folgt die Rekursion:

$$\delta_{s\bullet}(e) = 1 + \delta_{s\bullet}(f),\ e \in \mathscr{P}_s(f). \tag{3.24}$$

Der $\delta_{s\bullet}$-Wert für den Vorgänger $\{u,v\}$ einer Kante $\{v,w\}$ ist also um eins höher, da dieser Vorgänger auf allen kürzesten Wegen zwischen zwei Knoten s und t liegt, auf denen auch die Kante $\{v,w\}$ selbst liegt, aber zusätzlich auf dem kürzesten Weg von s nach v.

Es ist noch der Fall zu klären, dass $e = \{u,v\}$ keinen Vorgänger hat, also gilt $\nexists f :\ e \in \mathscr{P}_s(f)$. In diesem Fall ist v ein Blatt $(\deg(v) = 1)$ und e liegt genau für $t = v$ auf dem kürzesten Weg von s nach t. Damit ergibt sich:

$$\delta_{s\bullet}(\{u,v\}) = \delta_{sv}(\{u,v\}) = 1, \nexists f : e \in \mathscr{P}_s(f). \tag{3.25}$$

Somit lautet die vollständige Berechnungsvorschrift für $\delta_{s\bullet}$ für einen Weggraphen:

$$\delta_{s\bullet}(e) = \begin{cases} 1 + \delta_{s\bullet}(f) & e \in P_s(f) \\ 1 & \nexists f : e \in P_s(f). \end{cases} \tag{3.26}$$

Verallgemeinerung vom Weggraphen zum Baum

Im Weiteren sei der Graph $G = (V,E)$ ein Baum. Sollte eine Kante $e = \{u,v\} \in E$ nur Vorgänger einer einzigen Kante f sein, d.h. es gibt genau eine Kante $f \in E$, für die $f \in E : e \in \mathscr{P}_s(f)$ gilt, so bleibt die Beziehung aus Gleichung (3.24) erhalten.

Sollte es aber mehrere Kanten $f \in E$ geben, für die $e \in \mathscr{P}_s(f)$ gilt, also sich der Graph G in v in mehrere Äste aufgabeln, liegen die darauffolgenden Kanten f nur auf den kürzesten Wegen von s nach t, für die t von v aus auf dem selben Ast liegt wie f. Die Kante $e = \{u,v\}$ liegt ebenfalls auf allen dieser Wege und zusätzlich auf einem weiteren kürzesten Weg für $t = v$. Dieser Fall ist in Abbildung 3.6 dargestellt.

In diesem Fall gilt also:

$$\delta_{st}(e) = \begin{cases} 1 & \text{wenn } [\exists f \in E : e \in \mathscr{P}_s(f) \wedge \delta_{st}(f) = 1] \vee [t = v] \\ 0 & \text{sonst.} \end{cases} \tag{3.27}$$

Also gilt nun:

$$\delta_{s\bullet}(e) = \begin{cases} 1 & \nexists f : e \in \mathscr{P}_s(f) \\ 1 + \sum\limits_{f:\ e \in \mathscr{P}_s(f)} \delta_{s\bullet}(f) & \text{sonst,} \end{cases} \tag{3.28}$$

Rekusionsformel für beliebige Graphen

Der allgemeine Fall der Rekursionsformel zur Berechnung der akkumulierten Kanten-Betweenness für beliebige Graphen ist in Abbildung 3.7 dargestellt. Es gilt folgender Satz:

Satz 3 (Newman) *Sei* $(G = V,E)$ *ein Graph. Seien* $e = \{u,v\}$ *und* $f = \{v,w\}$, $e,f \in E$

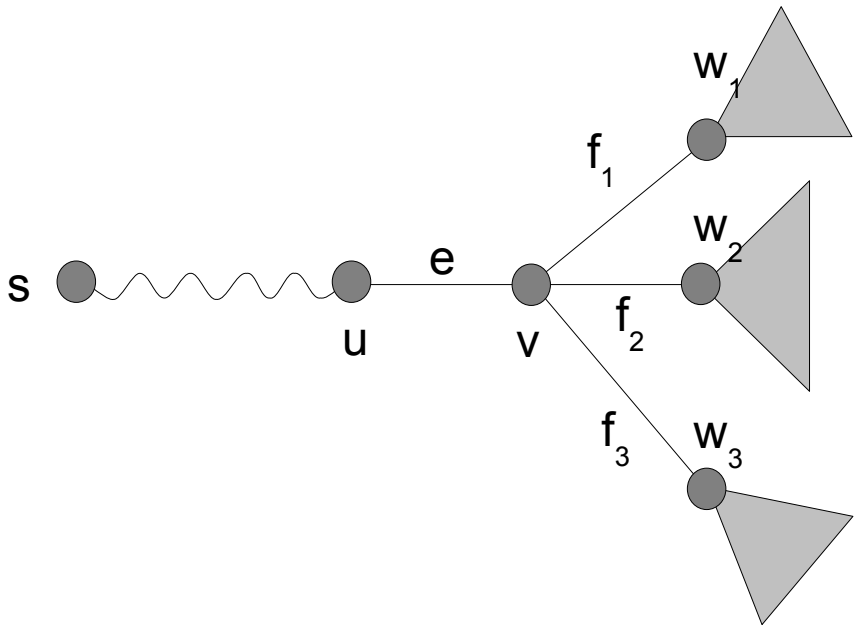


Abbildung 3.6: Akkumulierte Kanten-Paar-Betweenness für einen Baum. Alle kürzesten Wege, die über die Kanten f_1, f_2, f_3 laufen, laufen auch über e. Somit ist $\delta_{st}(e)$ für alle die t gleich $\delta_{st}(f_i)$, für die gilt: $d(s,t) \neq d(s,v)$. Für $d(s,t) = d(s,v)$ gilt: $\delta_{st}(e) = 1$ und $\delta_{st}(f_i) = 0$.

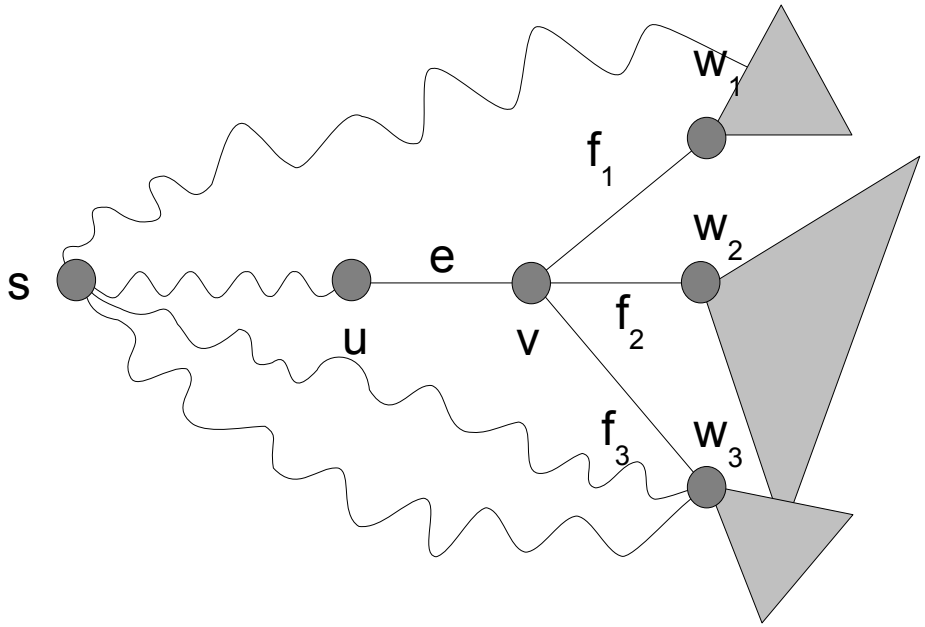


Abbildung 3.7: Allgemeiner Fall für die akkumulierte Kanten-Paar-Betweenness. Es kann kürzeste Wege von s nach v, von s zu einem der Knoten w_i und auch zu einem anderen Knoten t mit $d(s,t) > d(s,v)$ geben, die nicht über $\{u,v\}$ laufen.

und $s,t \in V$. Sei $d(s,u) < d(s,v)$ Dann gilt:

$$\delta_{s\bullet}(e) = \frac{\sigma_{su}}{\sigma_{sv}} \left(1 + \sum_{f:\ e \in \mathscr{P}_s(f)} \delta_{s\bullet}(f) \right).$$

Dieser Satz wurde von Newman in [7] nicht explizit formuliert, findet aber beim Algorithmus 3 Anwendung.

Der folgende Beweis orientiert sich am Beweis der für die Knoten-Betweenness geltenden Rekursion nach Brandes aus [6].

Beweis:
Es gelten die Bedingungen aus Satz 3. Dann gilt:

$$\begin{aligned}
\delta_{s\bullet}(e) &= \sum_{t \in V} \delta_{st}(e) && (3.29)\\
&= \sum_{\substack{t \in V \\ d(s,t) > d(s,v)}} \delta_{st}(e) + \sum_{\substack{t \in V \\ d(s,t) = d(s,v)}} \delta_{st}(e) + \sum_{\substack{t \in V \\ d(s,t) < d(s,v)}} \delta_{st}(e) && (3.30)\\
&= \sum_{\substack{t \in V \\ d(s,t) > d(s,v)}} \delta_{st}(e) + \delta_{sv}(e) + 0. && (3.31)
\end{aligned}$$

Führen wir eine weitere Form der Knoten-Paar-Betweenness analog zu [6] ein.

Definition 34 *Sei $G = (V,E)$ ein Graph und seien $e = \{u,v\}$,$f = \{v,w\}$, $e,f \in E$ und $s,t \in V$. Dann wird definiert:*

$$\sigma_{st}(e,f) := |\{P :\ P \text{ ist kürzester Weg zwischen } s \text{ und } t\ \wedge uevfw \leq P\}|$$

und

$$\delta_{st}(e,f) = \frac{\sigma_{st}(e,f)}{\sigma_{st}}.$$

Dann gilt für Formel 3.31:

$$\begin{aligned}
\delta_{s\bullet}(e) &= \sum_{\substack{t \in V \\ d(s,t) > d(s,v)}} \delta_{st}(e) + \delta_{sv}(e) && (3.32)\\
&= \sum_{\substack{t \in V \\ d(s,t) > d(s,v)}} \sum_{f:\ e \in \mathscr{P}_s(f)} \delta_{st}(e,f) + \delta_{sv}(e) && (3.33)\\
&= \sum_{f:\ e \in \mathscr{P}_s(f)} \sum_{\substack{t \in V \\ d(s,t) > d(s,v)}} \delta_{st}(e,f) + \delta_{sv}(e) && (3.34)
\end{aligned}$$

Jeder kürzeste Weg zwischen s und t mit $d(s,t) > d(s,v)$ läuft über eine Kante f, für die gilt $e \in \mathscr{P}_s(f)$. Somit werden mit dem Summieren über alle derartigen Kanten f für $\delta_{st}(e,f)$ wieder alle kürzesten Wege gezählt, die auch schon in $\delta_{st}(e)$ gezählt wurden. Folgende Beziehungen gelten aufgrund kombinatorischer Überlegungen:

$$\sigma_{st}(e,f) = \sigma_{su} \cdot \sigma_{wt} \tag{3.35}$$

und

$$\sigma_{st}(f) = \sigma_{sv} \cdot \sigma_{wt} \rightarrow \sigma_{wt} = \frac{\sigma_{st}(f)}{\sigma_{sv}}. \tag{3.36}$$

Durch Einsetzen von Gleichung (3.36) in Gleichung (3.35) erhält man:

$$\sigma_{st}(e,f) = \frac{\sigma_{su} \cdot \sigma_{st}(f)}{\sigma_{sv}}. \tag{3.37}$$

In die Definition von $\delta_{st}(e,f)$ eingesetzt, ergibt dies:

$$\delta_{st}(e,f) = \frac{\sigma_{su}}{\sigma_{sv}} \cdot \frac{\sigma_{st}(f)}{\sigma_{st}}. \tag{3.38}$$

Setzt man diese Darstellung für $\delta_{st}(e,f)$ in Zeile (3.34) ein, erhält man:

$$\delta_{s\bullet}(e) = \sum_{f:\ e \in \mathscr{P}_s(f)} \sum_{\substack{t \in V \\ d(s,t) > d(s,v)}} \delta_{st}(e,f) + \delta_{sv}(e) \tag{3.39}$$

$$= \sum_{f:\ e \in \mathscr{P}_s(f)} \sum_{\substack{t \in V \\ d(s,t) > d(s,v)}} \frac{\sigma_{su}}{\sigma_{sv}} \cdot \frac{\sigma_{st}(f)}{\sigma_{st}} + \delta_{sv}(e) \tag{3.40}$$

$$= \frac{\sigma_{su}}{\sigma_{sv}} \sum_{f:\ e \in \mathscr{P}_s(f)} \sum_{\substack{t \in V \\ d(s,t) > d(s,v)}} \frac{\sigma_{st}(f)}{\sigma_{st}} + \delta_{sv}(e) \tag{3.41}$$

$$= \frac{\sigma_{su}}{\sigma_{sv}} \sum_{f:\ e \in \mathscr{P}_s(f)} \delta_{s\bullet}(f) + \delta_{sv}(e). \tag{3.42}$$

Die Summation in Zeile (3.41) kann auch über alle $t \in V$ durchgeführt werden, da gilt:

$$\delta_{st}(f) = 0,\ d(s,t) \leq d(s,v). \tag{3.43}$$

Wegen:

$$\delta_{sv}(e) = \frac{\sigma_{su} \cdot \sigma_{vv}}{\sigma_{sv}} = \frac{\sigma_{su}}{\sigma_{sv}} \tag{3.44}$$

gilt weiter:

$$\delta_{s\bullet}(e) = \frac{\sigma_{su}}{\sigma_{sv}} \sum_{f:\, e \in \mathscr{P}_s(f)} \delta_{s\bullet}(f) + \delta_{sv}(e) \tag{3.45}$$

$$= \frac{\sigma_{su}}{\sigma_{sv}} \left(1 + \sum_{f:\, e \in \mathscr{P}_s(f)} \delta_{s\bullet}(f) \right). \tag{3.46}$$

□

Eine Kante {x,y} mit $d(s,x) = d(s,y)$ liegt nie auf einem kürzesten Weg zwischen s und irgendeinem Knoten t. Für sie gilt:

$$\delta_{s\bullet}(\{x,y\}) = 0. \tag{3.47}$$

Eine solche Kante ist für keine andere Kante Vorgänger.

3.5 Algorithmus zur Berechnung der Kantenbetweenness nach Newman

In [7] wird ein Algorithmus zur Berechnung der Kantenbetweenness unter Nutzung der Rekursion aus Abschnitt 3.4.4 vorgestellt. Ein beinahe identischer Algorithmus für die Berechnung der Knotenbetweenness unter Nutzung der Rekursion aus Abschnitt 3.4.3 wurde von Brandes in [6] vorgestellt.

Der Input für den Algorithmus ist ein zusammenhängender Graph $G = (V,E)$. Die Berechnungen werden für einen festen Startknoten s in zwei Stufen durchgeführt. In der ersten Stufe wird der Abstand aller Knoten zu einem Startknoten s berechnet. Dabei wird auch die Anzahl der Wege nach Lemma 4 ermittelt, die einen Knoten t von diesen Knoten s aus erreichen.

In der zweiten Stufe wird mithilfe der in Abschnitt 3.4.4 hergeleiteten Rekursion und unter Verwendung der Ergebnisse der ersten Stufe für alle Kanten e die akkumulierte Kanten-Paar-Betweeness $\delta_{s\bullet}(e)$ berechnet.

Diese zwei Stufen müssen für jeden Startknoten s separat durchgeführt werden.

Die erste Stufe basiert auf einer Breitensuche. Dabei muss neben der Entfernung aller Knoten $v \in V$ zu einem Startknoten s auch die Anzahl aller kürzesten Wege von s nach v berechnet werden. Es muss eine FIFO-Warteschlange Q initialisiert sein. In [7] wird mit ω_v die Anzahl σ_{sv} der kürzesten Wege, die von s aus v erreichen, bezeichnet. Mit d_v bezeichnet Newman den Abstand $d(s,v)$ des Knotens v zum Startknoten s. Entsprechend

gilt im Folgenden:

$$d_v = d(s,v) \text{ und}$$
$$\sigma_v = \sigma_{sv}.$$

Es werden die folgenden Schritte der Reihenfolge nach durchgeführt:

Algorithmus 3 (Berechnung der Kantenbetweenness nach [7])
Stufe 1:

1. *Setze $d_s = 0$ und $\sigma_s = 1$*
2. *Für jeden Knoten u aus der Nachbarschaft von s (ohne s selbst) wird gesetzt: $d_u = d_s + 1 = 1$ und $\sigma_u = \sigma_s = 1$. Danach werden alle Knoten u in die Warteschlange Q übergeben.*
3. *Nimm einen Knoten u aus der Warteschlange Q und tue für jeden Knoten v aus der Nachbarschaft von u folgendes:*

   a) *Wurde v noch kein Abstand d_v zugewiesen, ordne ihm $d_v = d_u + 1$ und $\sigma_v \leftarrow \sigma_u$ zu und gebe v in die Warteschlange Q.*
   b) *Wenn v bereits ein Abstand zugewiesen wurde und gilt $d_v = du + 1$, setzte $\sigma_v \leftarrow \sigma_v + \sigma_u$.*
   c) *Wenn v bereits ein Abstand zugewiesen wurde und gilt $d_v < d_u + 1$, tue nichts.*
4. *Wiederhole Schritt 3 bis Q leer ist und keine neuen Knoten mehr aufgenommen werden können.*

Nach Durchführung dieser Schritte wurde jedem Knoten in derselben Komponente ein Abstand zum Startknoten s und die Anzahl der kürzesten Wege von s nach t zugeordnet.

Die zweite Stufe verwendet die Rekursion aus Abschnitt 3.4.2. Es wird die Entfernung einer Kante zu einem Knoten benötigt:

Definition 35 *Sei $G = (V,E)$, $u,v,w \in V$ und $\{u,v\} \in E$. Dann ist der Abstand $d(w,\{u,v\})$ folgendermaßen definiert:*

$$d(w,\{u,v\} = \max(d(w,u),d(w,v)).$$

Es werden folgende Schritte ausgeführt:
Stufe 2:

1. *Für jedes Blatt t des Graphen und seinen Nachbarknoten u setze $\delta_{s\bullet}(\{u,t\}) = \sigma_u/\sigma_t$.*
2. *Finde die Kanten $\{u,v\}$ mit dem größten (endlichen) Abstand zu s. Jeder Kante*

$\{u,v\}$ mit $d_u < d_v$ wird zugewiesen:

$$\frac{\sigma_u}{\sigma_v}\left(1+\sum_{f:\ \{u,v\}\in\mathscr{P}(f)}\delta_{s\bullet}(f)\right)$$

3. *Wiederhole Schritt 2 für die Kanten mit dem nächst größten Abstand zu s, bis s erreicht ist.*

Stufe 1 und 2 sind für jeden Knoten $s \in V$ durchzuführen. Am Schluss sind die für jeden Startknoten s erhaltenen Werte aufzusummieren. Es gilt:

$$Z_B(e) = \sum_{s\in V}\delta_{s\bullet}(e). \tag{3.48}$$

Zur Verdeutlichung wurden die akkumulierten Knoten- und Kanten-Paar-Betweenness-werte für den Graphen aus Abbildung 3.8 berechnet. Die berechneten Werte sind in Tabelle 3.1 zu finden.

Da dieser Algorithmus nur auf Zusammenhangskomponenten anwendbar ist, muss er auf jede Komponente einzeln angewendet werden. Da in einem Schritt von einem Start-knoten s aus, hin zu allen anderen Knoten, alle akkumulierten Paar-Betweenness-Werte für alle Kanten e ermittelt werden, ermöglicht diese Rekursion die Berechnung eines Betweennessrankings für die Kanten eines Graphen in einer Zeit von $O(|V|(|V|+|E|))$ beziehungsweise $O(|V||E|)$ für $|V| << |E|$.

	$d(s,v)$	$\sigma_{sv} = \sum_{w \in \mathscr{P}_s(v)} \sigma_{sw}$	$\delta_{s\bullet}(v) = \sum_{w:\ v \in \mathscr{P}_s(w)} \frac{\sigma_{sv}}{\sigma_{sw}} (1 + \delta_{s\bullet}(w))$	$\delta_{s\bullet}(\{u,v\}) = \frac{\sigma_{su}}{\sigma_{sv}} \left(1 + \sum_{\{v,w\}:\ \{u,v\} \in \mathscr{P}_s(\{v,w\})} \delta_{s\bullet}(\{v,w\}\right)$
6	4	$= \sigma_{s4} + \sigma_{s5} = 4$	Per Def. $= 0$	
$(4,6),(5,6)$				$= \frac{2}{4}(0+1) = \frac{1}{2}$
$4,5$	3	$= \sigma_{s3} = 2$	$= \frac{2}{4} \cdot (1+0) = \frac{1}{2}$	
$(3,4),(3,5)$				$= \frac{2}{2}\left(1+\frac{1}{2}\right) = \frac{3}{2}$
3	2	$= \sigma_{s1} + \sigma_{s2} = 2$	$= \frac{2}{2}\left(1+\frac{1}{2}\right) + \frac{2}{2}\left(1+\frac{1}{2}\right) = 3$	
$(1,3),(2,3)$				$= \frac{1}{2}\left(1+\frac{3}{2}+\frac{3}{2}\right) = 2$
$1,2$	1	$= \sigma_{ss} = 1$	$= \frac{1}{2}(1+3) = 2$	
$(s,1),(s,2)$				$= \frac{1}{1}(1+2)$
s	0	Per Def. $= 1$	Per Def. $= 0$	

Tabelle 3.1: Beispiel für Algorithmus 3.

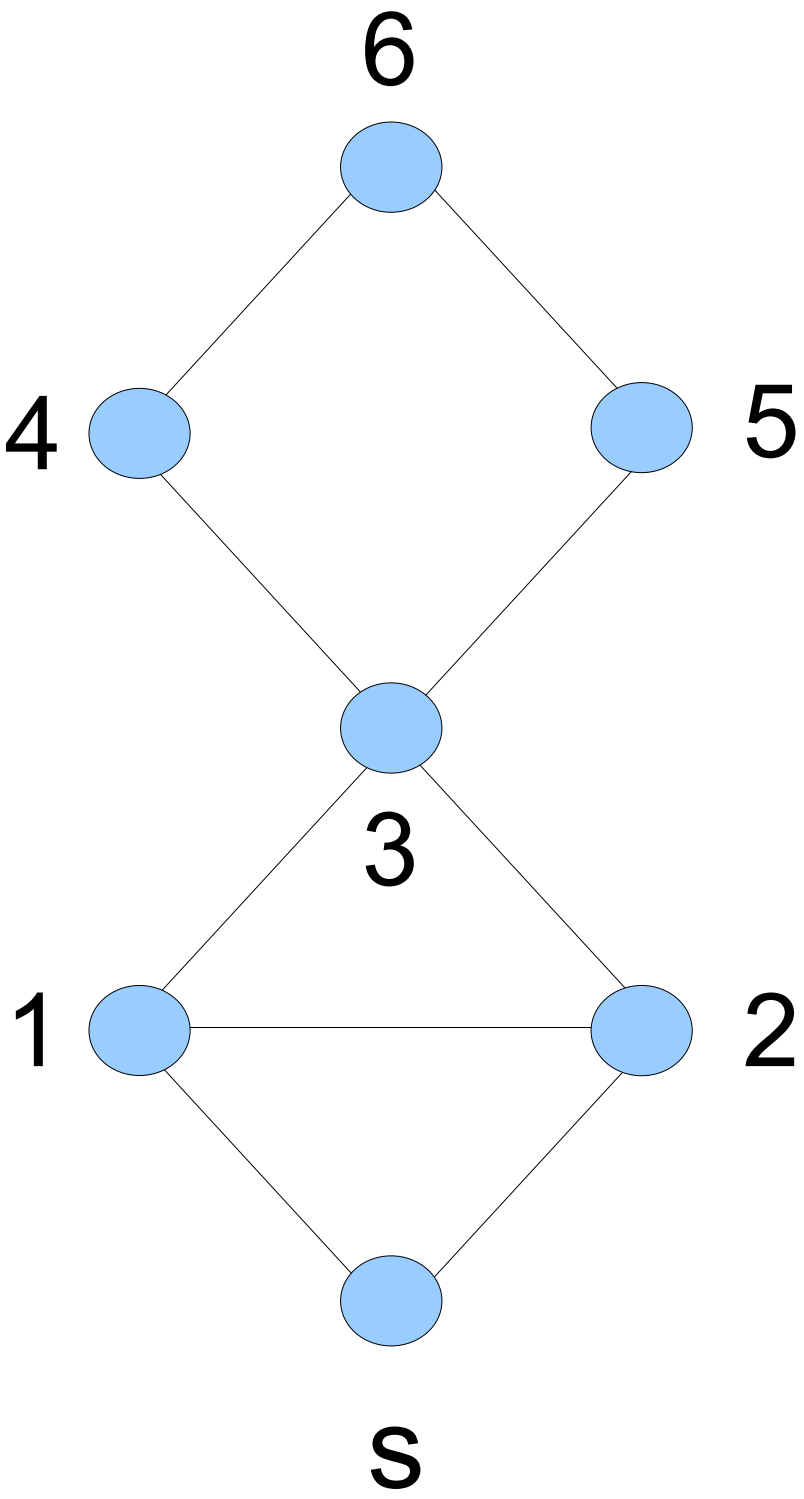


Abbildung 3.8: Beispiel Graph für Algorithmus 3.

3.6 Clusterung mittels Betweennessranking

Eines der Ziele dieser Arbeit ist es, Graphen mithilfe des Zentralitätsindex Kanten-Betweenness in Cluster zu zerlegen. Newman und Girvan haben in [7] einen Algorithmus vorgestellt, der genau das leistet:

Betrachten wir einen Graphen $G = (V,E)$ mit geclusterter Struktur. Sei C_i ein dichtes Cluster und s und t Knoten innerhalb dieses Clusters. Eine Kante $e \in E(C_i)$ kann nun auf vielen kürzesten Wegen zwischen s und t liegen. Allerdings gibt es dort wahrscheinlich auch viele kürzeste Wege zwischen s und t, die über andere Kanten als e verlaufen.

Betrachten wir nun einen Graphen $\hat{G}$ mit zwei dichten Clustern C_1 und C_2. Betrachten wir weiter eine Kante e, die als einzige Kante die Cluster C_1 und C_2 verbindet. Diese Kante muss automatisch auf **jedem** kürzesten Weg zwischen einem Knoten $s \in C_1$ und $t \in C_2$ liegen. Es sind also für Kanten, die dichte Bereiche eines Graphen miteinander verbinden, hohe Betweennesswerte zu erwarten.

Um nun einen Graphen in Cluster mit den gewünschten Eigenschaften aus Abschnitt 2.3 zu zerlegen, entfernt man die Kanten mit hohen Betweennesswerten. Die dabei entstehenden Komponenten werden mit den gesuchten Clustern identifiziert. Allerdings muss beachtet werden, dass die Betweennesswerte sich mit der Entfernung einer Kante ändern.

Sei $\tilde{G}$ ein Graph, hervorgegangen aus dem Graphen $\hat{G}$ durch Hinzufügen einer Kante f, die zusätzlich zur Kante e die Cluster C_1 und C_2 verbindet. Im Graphen $\tilde{G}$ laufen nicht mehr alle Kanten zwischen den Knoten $s \in C_1$ und $t \in C_2$ über die Kante e. Zumindest ein kürzester Weg zwischen den zu f inzidenten Knoten muss auch über f verlaufen.

Nehmen wir weiter an, fast alle der kürzesten Wege von Knoten in C_1 zu Knoten in C_2 verlaufen weiterhin über e und nur wenige über f. Entfernen wir nun e aufgrund seines hohen Betweennesswertes, müssen nun alle kürzesten Wege von C_1 nach C_2 über die Kante f verlaufen. Wir hatten aber für f nur einen niedrigen Betweennesswert ermittelt. Ein deutliches Beispiel für dieses Problem wird in Abbildung 3.9 dargestellt.

Es ist also notwendig, nach jeder Entfernung einer Kante e aus einem Graphen G die Betweennessberechnung erneut für den Graphen $G - e$ (das heißt für den Graphen $G' = (V, E \setminus \{e\})$), durchzuführen, bevor man die nächste Kante entfernt. Da zwischen den Komponenten des Graphen keine Wege und somit auch keine kürzesten Wege, verlaufen, ist die Neuberechnung nur bezüglich der Komponente notwendig, aus der die Kante e entfernt wurde.

Der Algorithmus wird abgebrochen, wenn der Graph in eine ausreichende Anzahl Komponenten zerfallen ist. Sollte eine solche Anzahl nicht durch das Problem bekannt sein,

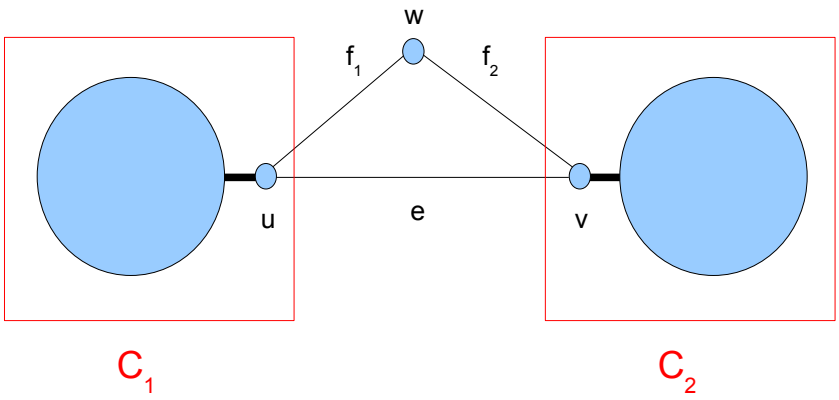


Abbildung 3.9: Das Cluster C_1, das den Knoten u enthält, ist mit dem Cluster C_2, das den Knoten v enthält, sowohl mit der Kante e als auch dem Weg uf_1wf_2v verbunden. Die Kanten f_1 und f_2 sind anfangs mit 1 bewertet. Nach der Entfernung von e laufen alle kürzesten Wege von C_1 nach C_2 über f_1 und f_2. Wird dann auch f_1 entfernt, ist f_2 wieder mit 1 bewertet.

müssen entweder Heuristiken Verwendung finden oder die gefundenen Clusterungen des Graphen bewertet werden.

Da maximal $m = |E|$ Kanten entfernt werden können, hat der Algorithmus eine Laufzeit von $O(m \cdot B)$, wobei B die benötigte Zeit zur Berechnung eines Kanten-Betweenness-Rankings ist. Wird dieses mit dem in Abschnitt 3.5 vorgestellten Algorithmus berechnet, ergibt sich für $|V| = n$ eine Gesamtlaufzeit von $O(mn(n+m)$ beziehungsweise von $O(nm^2)$ für $n << m$.

4 Praktische Auswertungen

Für die Berechnung und Bewertung der Clusterungen sowie für das Erstellen und Bewerten zufälliger geclusterter Graphen wurden im Rahmen dieser Bachelorarbeit implementierte Funktionen verwendet. Diese Funktionen inklusive einer Dokumentation befinden sich auf der CD im Anhang.

Desweiteren wurde die Pythonbibliothek ugraph [15] verwendet, die mir zur Durchführung dieser Arbeit von Herrn Professor Peter Tittmann zur Verfügung gestellt wurde.

Alle Berechnungen wurden mit Python 2.7.1 auf folgendem System durchgeführt:

- CPU: Intel Pentium 4 mit 1,99 GHz Taktfrequenz
- RAM: 512 MB
- Freier Festplattenspeicher: 19.5 GB
- Betriebssystem: Microsoft Windows XP Professional Version 2002 mit Service Pack 3

4.1 Die Graphen 3K7 und 7K4

Zum Testen des Newman-Girvan Algorithmus werden als erstes die Graphen 3K7 und 7K4 verwendet. Diese wurden bereits in [13] mithilfe einer spektralen Methode zerlegt. Dazu wird die Clusterzugehörigkeit der Knoten dargestellt über einen Cluster-Indikator-Vektor I der Form:

Definition 36 (Cluster-Indikator-Vektor) *Sei $G = (V,E)$ ein Graph und $v \in V$. Sei $C = \{C_1, \ldots, C_k\}$ eine Clusterung von G. Dann ist der Cluster-Indikator-Vektor $I = \{I_v\}_{v \in V}$ folgendermaßen definiert:*

$$I_v := l - 1,\ v \in C_l$$

Wenn der Algorithmus die erwartete Clusterung findet, dann müssen genau die Knoten denselben Eintrag im vom Algorithmus gefundenen Cluster-Indikator-Vektor zugeordnet bekommen, für die auch schon im erwarteten Cluster-Indikator-Vektor derselbe Eintrag zugeordnet wurde. Es müssen also die Cluster-Indikator-Vektoren identisch sein, bis auf eine Permutation der Clusternummer.

Für den Graph 3K7 aus Abbildung 4.1 ist bezüglich einer 3-Sektion die günstigste Clusterung (siehe Abbildung 4.2) intuitiv aufgrund der Struktur bekannt. In [13] wurde unter Verwendung einer Spektralmethode die intuitiv beste Clusterung in etwa der Hälfte

der Fälle gefunden. Nun soll überprüft werden, ob auch eine 3-Sektion mithilfe des Newman-Girvan-Algorithmus diese Clusterung erzeugt. Der Algorithmus erhält also die Information, dass er den Graph in drei Cluster zerlegen soll.

Wie in Tabelle 4.1 auf Seite 54 zu erkennen ist, erzeugt der Algorithmus von Newman und Girvan exakt die intuitiv erwartete Clusterung unter der Zusatzinformation, dass der Graph in drei Cluster zerlegt werden soll. Es wurden auch nur die erwarteten Inter-Cluster-Kanten entfernt. Allgemein ist beim Newman-Girvan-Algorithmus durchaus zu erwarten, dass auch vereinzelte Intra-Cluster-Kanten entfernt werden.

Der Graph 7K4 aus Abbildung 4.3 konnte von der Spektralmethode aus [13] nicht auf die erwartete Art und Weise geclustert werden.

Wie in Tabelle 4.2 auf Seite 54 ersichtlich, wurde unter Verwendung des Newman-Girvan-Algorithmus exakt die erwartete Clusterung gefunden. Die gleiche Benennung der Cluster ist auf die Implementation der Funktion zum Ermitteln der Komponenten zurückzuführen und allgemein nicht zu erwarten. Diese schreibt die Komponenten geordnet, beginnend mit der Komponente, die den Knoten mit der kleinsten Knotennummer beinhaltet, in ein Array. Die Clusternummern werden dann nach der Reihenfolge der Komponenten vergeben. Eine vergleichbare Sortierung wurde bei der Vergabe der Nummern für die erwarteten Cluster durchgeführt.

4.2 Test des Newman-Girvan-Algorithmus für zufällig erzeugte geclusterte Graphen

Es wurden Zufallsgraphen anhand des Zufallsalgorithmus aus Abschnitt 2.6 so erzeugt, dass die Clusterstruktur bis auf zufällige Abweichungen bekannt ist. Es wird im Folgenden untersucht, ob diese bekannten Clusterungen, im folgenden *erwartete Clusterung* genannt, vom Newman-Girvan-Algorithmus gefunden werden.

4.2.1 Erklärung zu den verwendete Zufallsgraphen

Bezüglich der Beschreibung der Zufallsgraphen wird für jeden Graph die Knotenmenge mit V und die Kantenmenge mit E bezeichnet. Zum Zweck der Beschreibung der Zufallsgraphen wird C als die aufgrund der Erzeugung des Graphen zu erwartende Clusterung angenommen. Intra-Cluster-Kanten werden mit einer Wahrscheinlichkeit von p erzeugt und Inter-Cluster-Kanten mit einer Wahrscheinlichkeit q. I ist der Cluster-Indikator-Vektor der erwarteten Clusterung. Die statistische Dichte der Cluster ist mit

$$\frac{|E(C_i)|}{|\binom{|C_i|}{2}|} \tag{4.1}$$

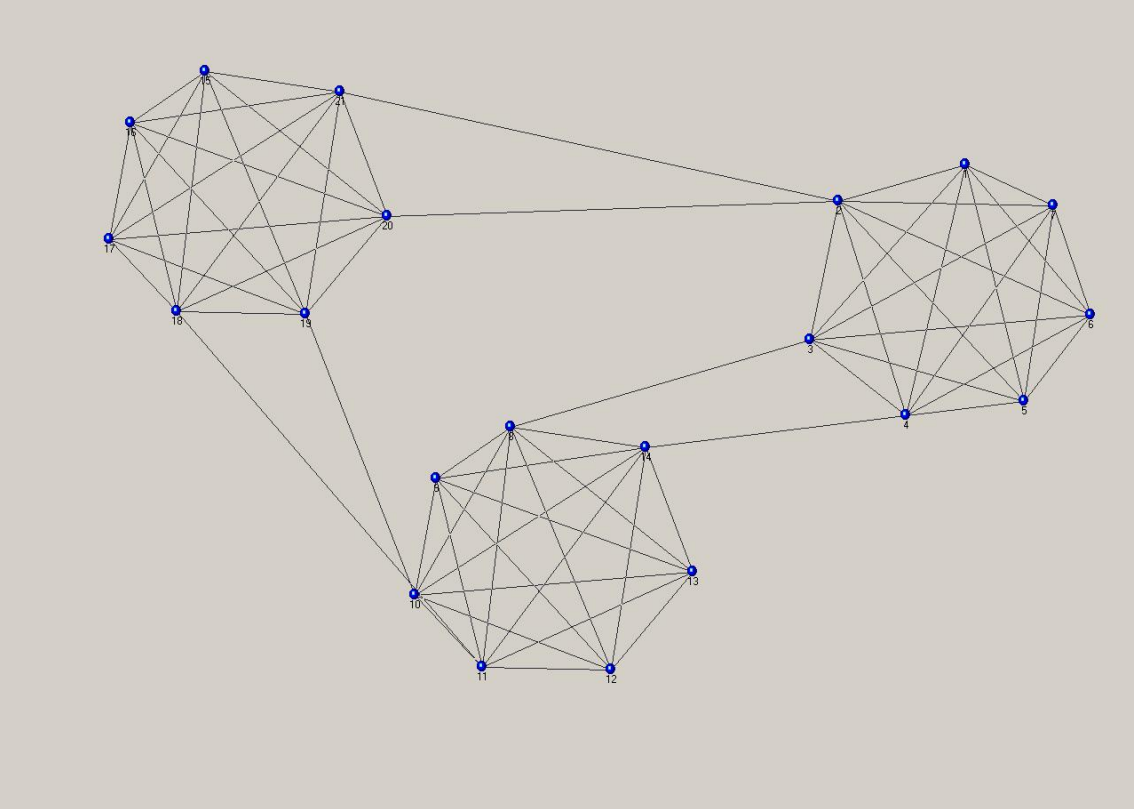


Abbildung 4.1: Graph 3K7: Drei vollständige Graphen der Ordnung 7 sind lose miteinander verbunden.

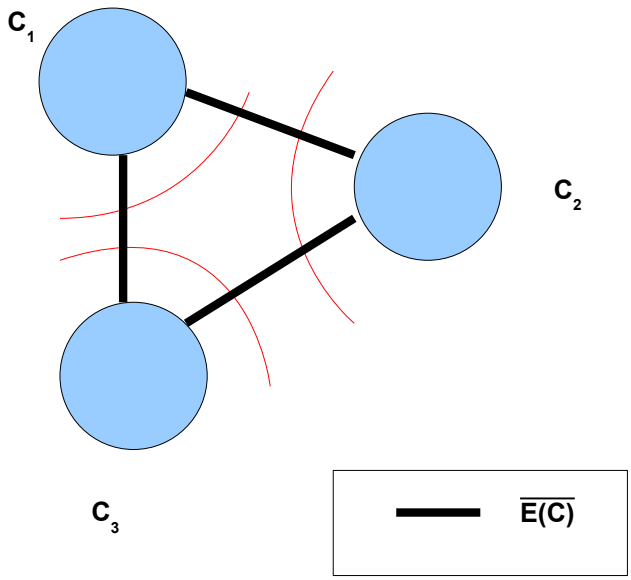


Abbildung 4.2: Graph 3K7: Intuitiv wird der Graph so zerlegt, dass jeder der Vollständigen Untergraphen ein Cluster bildet.

Cluster-Indikator-Vektor	
erwartet	0 0 0 0 0 0 0 1 1 1 1 1 1 1 2 2 2 2 2 2 2
berechnet	0 0 0 0 0 0 0 1 1 1 1 1 1 1 2 2 2 2 2 2 2
entfernte Kanten	$(2,21),(10,19),(4,14),(2,20),(3,8),(11,18)$

Tabelle 4.1: Graph 3K7: Auswertungen

Cluster-Indikator Vektor																												
erwartet	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4	5	5	5	5	6	6	6	6
berechnet	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4	5	5	5	5	6	6	6	6
entfernte Kanten	(2, 7), (12, 15), (4, 9), (7, 26), (20, 21), (11, 28), (3, 8), (12, 18), (6, 25), (16, 19), (17, 22), (5, 10), (12, 23), (13, 20)																											

Tabelle 4.2: Graph 7K4: Auswertungen

Cluster-Indikator Vektor																												
erwartet	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4	4
berechnet	0	0	0	0	0	0	0	0	0	0	1	0	2	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4	4

Tabelle 4.3: Random Graph 1: Auswertungen 1

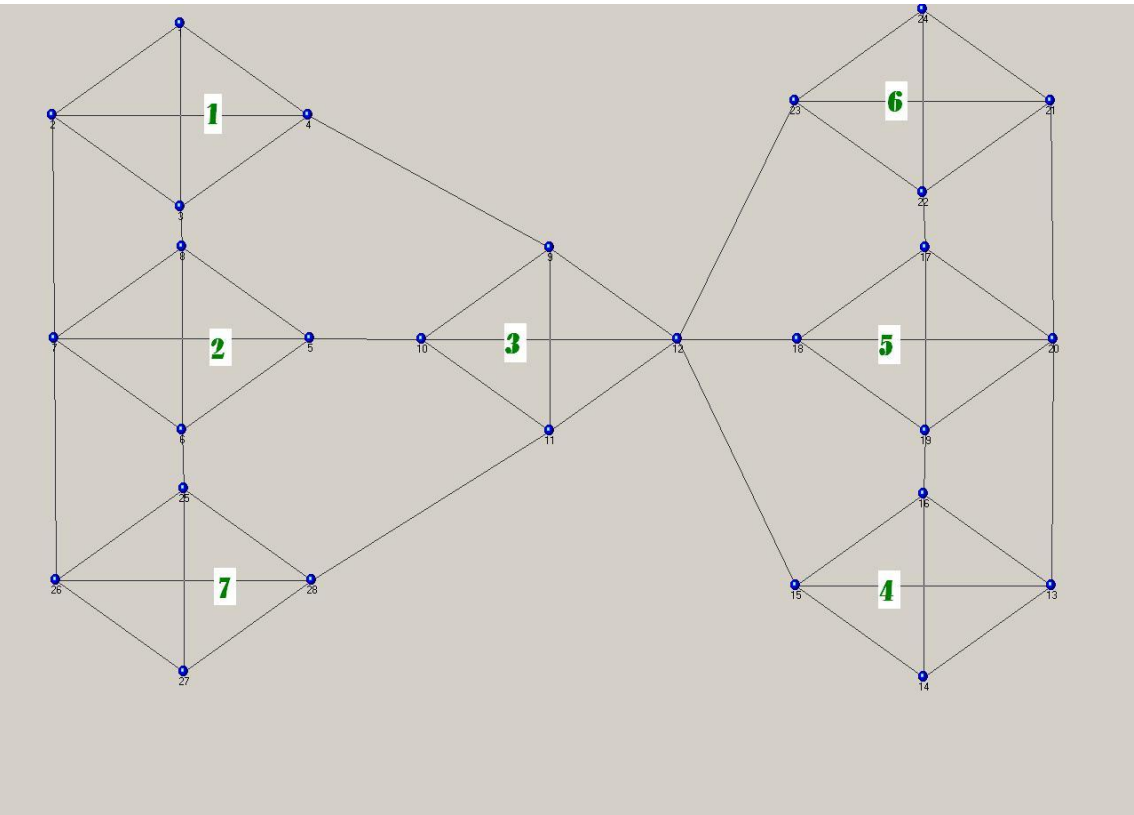


Abbildung 4.3: Graph 7K4: Sieben vollständige Graphen der Ordnung 4 sind zu einer symmetrischen Struktur verbunden. Der Knoten 12 ist eine Artikulation und sowohl den anderen Knoten des erwarteten Clusters 3 als auch nach außerhalb des Clusters 3 mit je 3 Kanten Verbunden.

für das Cluster $C_i \in C$ gegeben. Die statistische Dichte außerhalb der Cluster wird mit

$$\frac{|\overline{E(C)}|}{\binom{|V|}{2} - \sum_{C_i \in C} \binom{|C_i|}{2}} \tag{4.2}$$

erzeugt, wobei $\binom{|V|}{2} - \sum_{C_i \in C} \binom{|C_i|}{2}$ die mögliche Anzahl für Intra-Cluster-Kanten ist. Sie entspricht der Anzahl der Intra-Cluster-Kanten für den vollständigen Graphen $K_{|V|}$ mit der Knotenmenge V. Die verwendeten Zufallsgraphen befinden sich im Anhang. Die Knotenmenge, Kantenmenge, sowie die erwarteten wie erhaltenen Clusterungen sind in der Datei *RandomGraphs.txt* auf der CD im Anhang zu finden.

4.2.2 Random Cluster Graph 1

Im folgenden wird der Graph Random Cluster Graph 1 untersucht. Angaben zum Graph sind in der folgenden Tabelle zu finden.

Random Cluster Graph 1										
$\|V\|$	30									
$\|C\|$	5									
p	0.8									
q	0.05									
I	v	$1,\ldots,6$		$7,\ldots,12$		$13,\ldots,18$		$19,\ldots,24$		$25,\ldots,30$
	I_v	0		1		2		3		4
$\|E(C)\|$	66									
$\frac{\|E(C_i)\|}{\|\binom{\|C_i\|}{2}\|}$	C_i	0	1	2	3	4				
	Dichte	0.8	0.8	1.0	0.93	0.86				
$\|\overline{E(C)}\|$	28									
$\frac{\|\overline{E(C)}\|}{\binom{\|V\|}{2}-\sum_{C_i\in C}\binom{\|C_i\|}{2}}$	0.077									
Coverage	0.7									
Performance	398									
Intra-Cluster-Cond.	0.43									

In Tabelle 4.3 auf Seite 54 wird der Cluster-Indikator-Vektor der vom Algorithmus erzeugten Clusterung $\tilde{C}$ mit dem erwarteten Cluster-Indikator-Vektor verglichen. Es wurde nicht exakt der erwartete Cluster-Indikator-Vektor gefunden. Stattdessen wurden die ersten beiden erwarteten Cluster C_1 und C_2, bis auf den Knoten 11 aus dem erwarteten Cluster C_2, zu einem Cluster zusammengelegt. Knoten 11 bildet ein eigenes Cluster. Die restlichen Cluster wurden, wie erwartet, gefunden. Der Knoten 11 hat in G (Random Cluster Graph 1) die Nachbarn 8, 10, 12, 18 und 24, ist also kein Blatt. Der Inter-Cluster-Conductance-Wert für die erzeugte Clusterung beträgt 0. In Bezug auf Conductance wird ein Schnitt, der nur einen einzelnen Knoten aus dem Graphen schneidet, immer mit 1 bewertet. Durch die Bildung des Maximas in der Definition der Inter-Cluster-Conductance wird nur das am schlechtesten aus dem Graph geschnittene Cluster berücksichtigt, in diesem Fall mit dem Conductance-Wert 1. Für Inter-Cluster-Conductance ist dieser Wert aber von 1 abzuziehen. Der Inter-Cluster-Conductance-Wert von 0 aus Tabelle 4.4 auf Seite 58 wird also durch das Cluster C_2 (mit dem Cluster-Indikator-Vektor-Eintrag 1) bestehend aus dem Knoten 11 erzeugt. Dieser Knoten hat Verbindungen in alle erwarteten Cluster bis auf das Cluster C_1 und ist somit auch intuitiv schwer einem Cluster zuzuordnen. Ob die anderen Cluster gut aus dem Graphen getrennt sind, kann mithilfe der Inter-Custer-Conductance nicht bewertet werden, da nur die Conductance für das am schlechtesten aus dem Graphen geschnittene Cluster in den Index eingeht. Dafür können die Werte $\text{Cond}(\text{Cut}(G,\{C_i,\overline{C_i}\}))$ herangezogen werden, die gegebenenfalls schon für die Berechnung der Inter-Cluster-Conductance verwendet wurden. Für die Tabellen 4.6 und 4.7 auf Seite 59 wurden diese Conductance-Werte ermittelt. Dabei ist ersichtlich, dass abgesehen von dem erhalten Cluster $\tilde{C}_2$,

das nur aus dem Knoten 11 besteht, alle Cluster gut aus dem Graphen geschnitten sind. Ohne den vom Cluster $\tilde{C_2}$ erzeugten Conductance-Wert von 1 wäre die Inter-Cluster-Conductance sogar mit einem Wert von 0.58 besser, als die der erwarteten Clusterung. Die Zusammenfassung der erwarteten Cluster C_1 und C_2 ohne den Knoten 11 zum Cluster $\tilde{C_1}$ erzeugt dabei den günstigsten Schnitt. Die Cluster C_1 und C_2 aus der erwarteten Clusterung erzeugen zur Ermittlung von InterCC(G,C) die höchsten Conductance-Werte. Somit gibt es bezüglich einer guten Trennung der Cluster aus dem restlichen Graph strukturelle Gründe für die vom Newman-Girvan-Algorithmus gefundene Clusterung. In Bezug auf Coverage ist die vom Algorithmus gefundene Clusterung besser. Allerdings sind die Werte für Performance für die vom Newman-Girvan Algorithmus gefundene Clusterung schlechter, als die der erwarten Clusterung. Aufgrund dieser Ergebnisse wurde das weitere Zerfallen des Graphen für eine größere Clusterzahl untersucht.

Wie in Tabelle 4.4 zu sehen ist, ergibt auch eine weitere Zerlegung des Graphen keine bessere Clusterung. Beim Zerlegen des Graphen in 7 Cluster wird ersichtlich, dass der Knoten 2 aus Cluster C_1 mit den Knoten 7, 8 und 9 aus Cluster C_2 stark verbunden zu sein scheint, da sie in ein eigenes Cluster eingeordnet wurden. Tatsächlich existiert von jedem dieser Knoten eine Kante zu Knoten 2. Nach Entfernung dieser Kanten ergibt sich die Clusterung aus Tabelle 4.5. Die Cluster C_1 und C_2 wurden nun mit Ausnahme der Zuordnung des Knotens 11 gefunden. Es gibt also Abweichungen von der vorgesehenen und somit erwarteten Struktur des Graphen Random Cluster Graph 1. Die von der erwarteten Clusterung abweichende Clusterung, die mithilfe des Newman-Girvan-Algorithmus gefunden wurde, scheint durch diese abweichende Struktur begründet.

4.3 Empirische Laufzeit- und Exaktheitstests

4.3.1 Zeitmessung

Zur Zeitmessung wurde die Funktion *timeit* der Python Standard Bibliothek verwendet. Diese liest die Verweildauer eines Prozesses im System (WallClockTime [12]). Dabei wird auch die Rechenzeit anderer Prozesse mit einbezogen, die während dieser Zeit Prozessorzeit erhalten haben. Deshalb empfiehlt es sich, die Messungen mehrfach durchzuführen. Für deterministische Programme ist dann die kleinste gemessene Zeit die beste Schätzung für die Laufzeit, da eine längere als diese Laufzeit nur von programmfremden Prozessen verursacht worden sein kann.

Cluster-Indikator-Vektor																														Cov	Perf	InterCC
erwartet	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4	4	0.70	398	0.43
k[a]																																	
5	0	0	0	0	0	0	0	0	0	0	1	0	2	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4	4	0.77	385	0.0
6	0	0	0	0	0	0	0	0	0	1	2	0	3	3	3	3	3	3	4	4	4	4	4	4	5	5	5	5	5	5	0.73	389	0.0
7	0	1	0	0	0	0	1	1	1	2	3	1	4	4	4	4	4	4	5	5	5	5	5	5	6	6	6	6	6	6	0.64	396	0.0
8	0	1	0	0	0	0	1	1	1	2	3	1	4	4	4	4	4	4	5	5	5	5	5	5	6	6	6	6	6	7	0.60	393	0.0
9	0	1	2	2	2	2	1	1	1	3	4	1	5	5	5	5	5	5	6	6	6	6	6	6	7	7	7	7	7	8	0.56	391	0.0
10	0	1	2	2	2	2	3	3	3	4	5	3	6	6	6	6	6	6	7	7	7	7	7	7	8	8	8	8	8	9	0.53	389	0.0

Tabelle 4.4: Random Graph 1: Auswertungen 2

Cluster Indikator Vektor nach Entfernen der Kanten {2,7},{2,8},{2,9}																														Cov	Perf	InterCC
0	0	0	0	0	0	1	1	1	1	2	1	2	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4	4	0.70	396	0.39

Tabelle 4.5: Random Graph 1: Auswertungen 3

[a] Für Algorithmus vorgegebene Cluster Anzahl

Cond(Cut($G, \{\tilde{C}_i, V \setminus \tilde{C}_i\}$))				
$\tilde{C}_1$	$\tilde{C}_2$	$\tilde{C}_3$	$\tilde{C}_4$	$\tilde{C}_5$
0.27	1.0	0.42	0.39	0.38

Tabelle 4.6: Random Cluster Graph 1: Auswertungen 4

Cond(Cut($G, \{C_i, V \setminus C_i\}$))				
C_1	C_2	C_3	C_4	C_5
0.5	0.57	0.42	0.39	0.38

Tabelle 4.7: Random Cluster Graph 1: Auswertungen 5

4.3.2 Vergleich Clusteralgorithmus mit klassischer Betweennessberechnung und mit Betweennessberechnung nach Newman bzw. Brandes

Es wurde für eine Reihe von Zufallsgraphen die Laufzeit des im Rahmen dieser Bachelorarbeit implementierten Newman-Girvan-Algorithmus [7], der die Kanten-Betweenness-Werte nach dem Algorithmus von Newman [5] beziehungsweise Brandes [6] unter Verwendung der akkumulierten Paar-Betweenness berechnet (Implementation I) verglichen mit einer Implementation des Newman-Girvan-Algorithmus unter Verwendung einer klassischen Betweennessberechnung, welche Bestandteil der Pythonbibliothek ugraph [15] ist (Implementation II). Es wurde für die Graphen Random Cluster Graph 2 und Random Cluster Graph 3 bei 100 Testläufen die Zeit gemessen und überprüft, ob dieselben Kanten entfernt wurden. Die angegebenen Zeiten sind die kleinsten gemessenen Zeiten. Da die Minimallaufzeiten stets weniger als 10% kleiner waren, als die Maximallaufzeiten wurde für weitere Graphen nur ein einziger Durchlauf durchgeführt. Die Laufzeiten sind in Sekunden gegeben.

In Tabelle 4.8 ist zu erkennen, dass die Implementation I, wie erwartet, weniger Zeit benötigt, als Implementation II. Außerdem sind die erzeugten Clusterungen für alle getesteten Graphen identisch. Grundsätzlich besteht im Newman-Girvan-Algorithmus die Möglichkeit, dass mehr als nur eine Kante die höchste Betweennessbewertung erhält. Dann besteht die Wahlmöglichkeit, welche dieser Kanten entfernt wird. Deswegen müssen zwei verschiedene Implementierungen des Newman-Girvan-Algorithmus nicht zwingend immer die gleichen Clusterungen erzeugen. Für die Graphen Random Cluster Graph 2 bis 4 wurden die Clustergüteindizes Coverage, Performance und Inter-Cluster-Conductance ermittelt. Die Ergebnisse sind in Tabelle 4.9 erfasst. Für Random Cluster Graph 4 wurde die erwartete Clusterung nicht gefunden. Allerdings weichen sowohl der Cluster-Indikator-Vektor, als auch die berechneten Gütemaße von erwarteter und berechneter Clusterung, nur minimal voneinander ab. Es wurde nur ein einziger Knoten anders als erwartet zugeordnet. Die Clustergüteindizes für die erwarteten Clusterungen

sind in den Tabellen im Abschnitt A zu finden.

Es wurden Tests durchgeführt, in denen nur die Laufzeit für Implementation I gemessen wurde. Die Ergebnisse sind in Tabelle 4.10 ausgelistet. Es ist zu erkennen, dass die Laufzeit in hohem Maß von der Kantendichte des Graphen abhängig ist. Für ausführlichere Tests zur Laufzeit für große Graphen sind jedoch erst weiterführende Untersuchungen bezüglich der Parameter für zufällig erzeugte Graphen nach Abschnitt 2.6 oder alternativ Datensätze entsprechenden Umfangs für die Erzeugung geeigneter Graphen nach Abschnitt 2.2 erforderlich.

Graph	Laufzeit Implimentation I	Laufzeit Implementation II	Entfernte Kanten identisch?
Random Cluster Graph 2	1.26	9.26	*Ja*
Random Cluster Graph 3	21.72	292.83	*Ja*
Random Cluster Graph 4	54.67	740.79	*Ja*
Random Cluster Graph 5	358.41	5396.34	*Ja*

Tabelle 4.8: Laufzeiten des Newman-Girvan-Algorithmus 1

Graph	Coverage	Unnomalized Performance	Inter Cluster Conductance	erwarteter Indikatorvektor gefunden?
Random Cluster Graph 2	0.86	1194	0.67	*Ja*
Random Cluster Graph 3	0.88	4739	0.75	*Ja*
Random Cluster Graph 4	0.75	4679	0.47	*Nein*

Tabelle 4.9: Clustergüte für Random Cluster Graph 2 bis 4

Parameter Zufallsgraph				Laufzeit
$\|V\|$	$\|C\|$	p	q	
500	25	0.5	0.0005	2238.30
1000	50	0.3	0.00005	27.56
1000	50	0.3	0.0001	2808.57

Tabelle 4.10: Laufzeiten des Newman-Girvan-Algorithmus 2

5 Zusammenfassung und Ausblick

5.1 Zusammenfassung

Der Newman-Girvan-Algorithmen hat für die meisten untersuchten Graphen exakt die erwarteten Clusterungen gefunden, die Aufgrund der Art der Erzeugung der entsprechenden Graphen bekannt waren. Für Random Cluster Graph 1 ist eine entstandene Abweichung durch die Struktur des Graphen begründet und vertretbar. Die Abweichung für Random Cluster Graph 4 betraf nur einen einzigen Knoten und wurde nicht weiter untersucht. Die Laufzeit des Newman-Girvan-Custeralgorithmus war unter Verwendung des Algorithmus zur schnelleren Berechnung eines Betweennessrankings über akkumulierte Paar-Betweenness nach Newman [5] und unabhängig nach Brandes [6] (siehe Abschnitt 3.5) deutlich geringer als mithilfe der klassischen Variante. Die Erzeugung zufälliger geclusterter Graphen erwies sich für Knotenzahlen größer einhundert als schwierig. Grund dafür war die Wahl geeigneter Parameter. Oft war die Clusterstruktur zu eindeutig, das heißt, die erwarteten Cluster waren bereits beinahe separiert. In anderen Fällen wies der Graph zwischen den erwarteten Clustern eine größere Dichte auf, als innerhalb und besaß dementsprechend keine geclusterte Struktur.

5.2 Ausblick

Folgendes wird zur Weiterführung der Arbeit empfohlen:

1. Effizientere Implementation der Algorithmen:
   - Im Cluster-Algorithmus von Newman und Girvan müsste die Neuberechnung der Betweennesswerte nur für die Komponente des Graphen durchgeführt werden, aus der zuletzt eine Kante entfernt wurde. Diese Tatsache wurde noch nicht berücksichtigt.
   - In der Berechnung des Kanten-Betweenness-Rankings nach Newman erwies sich die Ansteuerung der Vorgängerkanten im Schritt 2 als schwer kalkulierbarer zusätzlicher Aufwand. Es sollte geprüft werden, ob diese Information schon im Schritt 1 gewonnen werden kann, zum Bespiel gespeichert als ein entsprechender gerichteter Graph, der nur die benötigten Kanten enthält.
2. Es sollten weitere Tests für große, geclusterte Graphen durchgeführt werden.
3. Für den Algorithmus zur zufälligen Erzeugung geclusterter Graphen sind Untersuchungen bezüglich der sinnvollen Parameterwahl notwendig, um sicherzustellen, dass eine geclusterte Struktur entsteht.

4. Es sollten weitere Gütemaße zur Bewertung der Clusterungen herangezogen werden, zum Beispiel Modularity [7].
5. Erweiterungen auf gerichtete und gerichtete Graphen.

Anhang A: Tabellen zu den Zufallsgraphen

Random Cluster Graph 1

$\|V\|$						30
$\|C\|$						5
p						0.8
q						0.05
I	v	$1,\ldots,6$	$7,\ldots,12$	$13,\ldots,18$	$19,\ldots,24$	$25,\ldots,30$
	I_v	0	1	2	3	4
$\|E(C)\|$						66
$\frac{\|E(C_i)\|}{\|\binom{\|C_i\|}{2}\|}$	C_i	0	1	2	3	4
	Dichte	0.8	0.8	1.0	0.93	0.86
$\|\overline{E(C)}\|$						28
$\frac{\|\overline{E(C)}\|}{\binom{\|V\|}{2}-\sum_{C_i\in C}\binom{\|C_i\|}{2}}$						0.077
Coverage						0.7
Unnorm. Performance						398
Intra-Cluster-Cond.						0.43

Tabelle A.1:

Random Cluster Graph 2

$\|V\|$											50
$\|C\|$											10
p											0.8
q											0.01
$\|E(C)\|$											82
$\frac{\|E(C_i)\|}{\|\binom{\|C_i\|}{2}\|}$	C_i	0	1	2	3	4	5	6	7	8	9
	Dichte	0.9	0.7	0.7	0.8	0.8	1.0	0.8	0.8	1.0	0.7
$\|\overline{E(C)}\|$											13
$\frac{\|\overline{E(C)}\|}{\binom{\|V\|}{2}-\sum_{C_i\in C}\binom{\|C_i\|}{2}}$											0.0116
Coverage											0.86
Unnorm. Performance											1194
Intra-Cluster-Cond.											0.67

Tabelle A.2:

Random Cluster Graph 3

$\|V\|$											100
$\|C\|$											10
p											0.6
q											0.005
$\|E(C)\|$											275
$\frac{\|E(C_i)\|}{\|\binom{\|C_i\|}{2}\|}$	C_i	0	1	2	3	4	5	6	7	8	9
	Dichte	0.6	0.67	0.64	0.6	0.62	0.67	0.49	0.67	0.56	0.69
$\|\overline{E(C)}\|$											36
$\frac{\|\overline{E(C)}\|}{\binom{\|V\|}{2} - \sum_{C_i \in C} \binom{\|C_i\|}{2}}$											0.008
Coverage											0.88
Unnorm. Performance											4739
Intra-Cluster-Cond.											0.75

Tabelle A.3:

Random Cluster Graph 4

$\|V\|$											100
$\|C\|$											10
p											0.6
q											0.01
$\|E(C)\|$											296
$\frac{\|E(C_i)\|}{\|\binom{\|C_i\|}{2}\|}$	C_i	0	1	2	3	4	5	6	7	8	9
	Dichte	0.53	0.62	0.64	0.40	0.62	0.44	0.73	0.6	0.73	0.64
$\|\overline{E(C)}\|$											87
$\frac{\|\overline{E(C)}\|}{\binom{\|V\|}{2} - \sum_{C_i \in C} \binom{\|C_i\|}{2}}$											0.019
Coverage											0.76
Unnorm. Performance											4682
Intra-Cluster-Cond.											0.47

Tabelle A.4:

Anhang B: CD

Die CD zur Bachelorarbeit ist im Buchrücken eingelegt. Sie besitzt folgenden Inhalt:

- Beispielgraphen
- PythonProgramme
- Dokumentation.pdf
- RandomGraphs.txt

Eine ausführliche Beschreibung des Inhalts und der für diese Arbeit verwendeten Programme findet sich in der Datei Dokumentation.pdf.

Literaturverzeichnis

[1] Ulrik Brandes, Thomas Erlebach: Network Analysis, *Springer Verlag Berlin Heidelberg* (2005)

[2] Th. H. Cormen, C. E. Leiserson, R. Rivest, C. Stein: Algorithmen-Eine Einführung, *Oldenbourg Verlag München Wien*, (2004)

[3] Elisa Schaeffer: Survey Graph Clustering. Laboratory for Theoretical Computer Science, Helsinki University of Technology TKK, P. O. Box 5400, FI02015 TKK, Finland

[4] M. E. J. Newman: Networks, an Introduction, *Oxford University Press* (2010)

[5] M. E. J. Newman: Scientific collaboration networks: II. Shortest paths, weighted networks, and centrality. *Phys. Rev. E* **64**, 016132 (2001)

[6] U. Brandes: A faster algorithm for betweenness centrality. *Journal of Mathematical Sociaty* **25**, 163-177 (2001)

[7] M. E. J. Newman, M. Girvan: Finding and evaluating community structure in networks.*Phys. Rev.* E **69**, 026113 (2004)

[8] L. Freeman: A set of measures of centrality based upon betweenness. *Sociometry* **40** 35-41 (1977)

[9] J. M. Anthonisse: The rush in a directed graph. Technical Report BN 9/71, Stichting Mathematicsh Centrum, Amsterdam (1971)

[10] J. Shi, J. Malik: Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 22(8),88-905 (2000)

[11] L. Hagen, A. Kahng: New spectral methods for ratio cut partitioning and clustering. *IEEE Trans. Comput.-Aided Des.* 11(9),1074-1085 (1992)

[12] http://www.python.org/doc, 15.01.2012

[13] Alrik Messner, Praktikumsbericht: Spectral Clustering for Simple and Undirected Graphs, Hochschule Mittweida, (2011)

[14] B.W. Kernighan, S. Lin: An efficent heuristic procedure for partitioning graphs, *Bell System Technical Jounal* **49** 291-307 (1970)

[15] Peter Tittmann: Pythonbibliothek ugraph.py, Hochschule Mittweida, (2012)

[16] Ulrike von Luxburg: A tutorial on spectral clustering. Stat Comput (2007)

[17] M.E.J Newman: Finding community structure using eigenvectors of matrices. Department of Physics and Center of Study of Complex Systems, University Michigan, Ann Arbor, MI 48109-1040 (2006)

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, 22. Februar 2012